THE HONG KONG POLYTECHNIC UNIVERSITY

DEPARTMENT OF COMPUTING

AI-TIMES: A PARALLEL WEB NEWS RETRIEVAL SYSTEM

BY

LUO WEIDONG

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIRMENTS FOR THE DEGREE OF

MASTER OF PHILOSOPHY

INITIAL SUBMISSION: JAN, 2007

Certificate of Originality

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

(Signed)

Name: <Luo Weidong>

Acknowledgements

I would like to acknowledge the partial support of the ICRG grant A-PF83 and Teaching Company Scheme between Hong Kong Polytechnic University and Ai-Times Company Ltd.

Special thanks to my supervisor, Dr. James N.K LIU, for his invaluable guidance, helpful advice and constructive criticisms. Without his insight, enthusiasm, encouragement and inspiration, I would not have the courage and determination to complete this work.

Above all, I thank to my dear parents for their sacrificial love and endless support.

Publications arising from the thesis

(1) James N.K. Liu, Weidong Luo and Edmond Chan (2007), Design and Implementation of A High Performance Distributed News Retrieval System, accepted by International Journal of Knowledge-based and Intelligent Engineering Systems.

(2) Manchung Chan, Weidong Luo and James N.K. Liu (2007) Design and Implementation of a High Performance Distributed News Retrieval System, in Proceedings of ICITM2007, January 3-5, 2007, Hong Kong, pp.26-33

(3) James Liu, Bo Feng, Meng Wang and Weidong Luo (2006) Tropical cyclone forecast using angle features and time warping, in Proceedings of IJCNN 06, the World Congress on Computational Intelligence (WCCI 06), July 16-21, 2006, Vancouver, BC, Canada.

(4) James N.K. Liu, Weidong Luo, Edmond M.C. Chan (2005) Design and Implement a Web News Retrieval System, Lecture Notes in Computer Science 3683, Springer pp. 149-156 (in Proceedings of Ninth International Conference on Knowledge-Based Intelligent Information & Engineering Systems (KES2005), September 14-16, 2005, Melbourne, Australia). http://dx.doi.org/10.1007/11553939_22

Abstract

The explosion in the availability of online information easily accessible through the Internet is a reality. As the available information increases, the inability to process, assimilate and use such large amount of information becomes more and more apparent. Online news information suffers from these problems.

Currently available web news retrieval systems face a number of problems in that web-based news retrieval requires the ability to quickly and accurately process and update very large amounts of data that is constantly being updated.

In this thesis, we present the design and implementation of Ai-Times, a parallel web news retrieval system the goal of which is to accurately retrieve and organize the web news information. This version of Ai-Times introduces the following novel algorithms: A novel optimized crawler algorithm whose fetching-speed is 6 times faster than that of the traditional crawler; A keen tag based extraction algorithm which can extract the data rich content with minimal manual effort and which also allows data to be classified as important or not important so that the crawler can revisit and update important data; A modified vector space model improved using query expansion and term reweighting and the most valuable contribution, an modified MapReduce improved by estimating the execution time of each subtask, which is proven to be able to reduce the number of the unusual tasks and shorten the whole job execution time.

Table of contents

List of Figures

List of Tables

Chapter 1

Introduction

The explosion in the availability of online information easily accessible through the Internet is a reality. As the available information increases, the inability to process, assimilate and use such large amounts of information becomes more and more apparent. Online news information suffers from these problems.

Web-based information retrieval (IR) systems fetch, extract, and index data from the web for end users. Such systems traditionally contain four modules: a crawler module, a data extraction module, an inverted index engine, and a retrieval module which are respectively responsible for fetching, extracting, indexing, and returning data for users. Current web-based news retrieval systems operate in this way but face a number of problems in that web-based news retrieval requires the ability to quickly and accurately process and update a very large amount of data that is constantly being updated.

First, web-based news is a very large and increasingly growing domain, making it very difficult to effectively crawl all of the many millions of news web pages now available to us. The modules especially the crawler module in a traditional IR system is not able to finish processing millions of web documents within a reasonable amount of time.

Second, news sites update their contents frequently so any effective web news IR system must also be able to update frequently, yet this is impossible given that current approaches revisit all of the web documents without distinction during the update interval.

Third, most web documents are semi-structured; that is, they do not all use a standard format or hierarchy that clearly signals the data-rich parts of the document - the valuable content most users are interested in. Consequently, current data rich part extraction algorithms cannot automatically distinguish between valuable and non-valuable data. This means that current IR systems must inefficiently refresh and return a large amount of data that is either not of interest to users or that is not frequently updated, such as advertisements. As result returns a large amount of irrelevant data, this is slow and unsatisfactory for user.

1.1 Objective

In order to accurately retrieve and organize the web news information better, we have designed and implemented Ai-Times- a web-based news retrieval system the goal of which is to accurately retrieve and organize the web news information. Compared with traditional news retrieval systems, given same resources and time, Ai-Times can fetch and process more news documents with less manual effort. At the first stage, we have designed and implemented a basic web-based news retrieval system with an optimized crawler algorithm [20]. The Ai-Times system contains some typical web based information retrieval system modules: an optimized crawler, a news extraction module using novel keen tag extraction algorithm, an inverted index engine, a retrieval module and an automatic summarization module. With these modules, Ai-Times can:

- Fetch the web pages from user-defined websites with a higher update frequency
- Extract the valuable content with less manual involvement than other systems
- Index and summarize the extracted news, and provide a friendly user interface for user to retrieve the news.

As the size of the web grows, it is difficult for a single node to finish downloading, extracting and indexing pages in a reasonable amount of time. In order to address this problem, we apply an improved MapReduce algorithm in Ai-Times to make Ai-Times a parallel system with following characteristics:

- The system can process millions of web news per day.
- The system is running on a number of machines and is scalable enough so that we can easily add or remove machines.
- The system has good fault tolerant ability. Failure of a running machine or an active hard disk should not cause the whole system fail or any data lost.

1.2 Background and Literature Review

In this section, firstly we review the previous work of the architecture of news information system, which will help us of the design of whole system. Then we survey the related work of data rich part extraction algorithm and crawler module. Finally we review the literatures of MapReduce based on which we propose the improved MapReduce.

1.2.1 News information retrieval

News information retrieval has been well studied [1, 2, 17, 22, 23, 24, 28]. Many studies have been done on the architecture of news information retrieval. For example, Ariki and Sugiyama [2] present a system that automatically classifies TV news articles using keywords; Renals et al. [24] describe the THISL news retrieval system maintaining an archive of BBC radio and televisions news records; Aggarwal and Hung [1] introduce the WIRE-a WWW-based information retrieval and extraction system; Sanderson and van Rijsbergen [28] present the design of a news retrieval toolsbased on an existing database of some newspapers such as Times.

Morrison and Jose [22] developed VideoSqueak, which records continuous news broadcasts and facilitates online querying and retrieval of the video archive created. Lim and Ng [19] developed DatAQs which analyzes, identifies, and categorizes languages used in HTML documents and extracts information from HTML documents of interest written in different languages.

The above literatures contribute much to the general architecture of news information retrieval. However, none of them describes the core modules such as the crawler module, the news extraction module and the automatically summarization module in detail.

1.2.2 Data-rich part extraction

In one typical approach that address the inability of current algorithms for extracting the data rich part of web content, Mukherjee et al. [23] proposed an algorithm to partition the html document into tree-like semantic structures. This approach, however, does not process all types of web pages well. In an earlier approach [4], the authors presented a novel keen tag based extraction algorithm. The keen tag based extraction algorithm proved to be effective on most html pages yet required some training.

1.2.3 Web Crawler

Boldi et al. [7] and Shkapenyuk and Suel [29] introduce some means of how to implement a web crawler. Many studies have been done on implementing a domain-specific crawler. Hu and Wong [16] presented a simple probabilistic model for intelligent Web crawlers. Bergholz and Childlovskii [5] developed a crawler starting from the PIW (Publicly Indexable Web) to find entry points into the hidden

Web. We presented an optimized web crawler [20] which is applicable to be used in news information retrieval system. Details are given in Chapter 3.

1.2.4 Parallel crawler

Many researches [7, 10, 11, 12, 29] have been done on improving the speed of crawler modules have adopted parallel approaches – both architectures and algorithms in parallel crawlers. A parallel crawler is a crawler whose fetching tasks are executed in parallel and therefore more efficiently on different machines. Buzzi [10] proposed a scheme to permit a parallel crawler to acquire information about the global state of a Website before the crawling process takes place. However, this scheme requires Web server cooperation in order to collect and publish information content which is needed for enabling a crawler to tune its visit strategy. The parallel crawler's performance is relatively poor when it is directly applied to a news retrieval system. A more general drawback of parallel approaches for news retrieval however is that they still require that the crawler revisits all of the documents rather than just those documents which are of interest to users or which are most likely to have been updated since the last visit. We presented an optimized parallel crawler [21], which is proved to have better performance than traditional parallel crawler.

1.2.5 MapReduce

In an approach which goes beyond paralleling just crawlers, Google is using the MapReduce [13] library to line up all modules in parallel in their system, partitioning tasks into many different subtasks. MapReduce is a programming model and an associated implementation for processing and generating large data sets. Many studies [6, 18] have been done on parallelizing the computation by using some sort of programming model. People are interested in MapReduce since it has been successfully used in Google. Some examples [32, 33, 34, 35, 36] of MapReduce are available on web. MapReduce has been highly effective; however, traditinoal MapReduce still suffers from the drawback that it is not able to estimate execution times for individual subtasks, so the shortest waiting time for the completion of any task could be the longest time of any sub-tasks. This makes the MapReduce fail in some special scenarios.

Chapter 2

Preliminaries

In this chapter we describe the fundamental theories and algorithms which will be used in Ai-Times.

2.1 Inverted Indexing

An inverted index is an index structure storing a mapping from words to their locations in a document or a set of documents, allowing full text search. It is the most popular data structure used in document retrieval systems. Given the texts T0 = "it is what it is", T1 = "what is it" and T2 = "it is a banana", we have the following inverted file index:

"a":	{(2, 2)}
"banana":	{(2, 3)}
"is":	{(0, 1), (0, 4), (1, 1), (2, 1)}
"it":	{(0, 0), (0, 3), (1, 2), (2, 0)}
"what":	{(0, 2), (1, 0)}

where the pairs are document numbers and local word numbers. Like the document numbers, local word numbers also begin with zero. So, "banana": {(2, 3)} means the word "banana" is in the third document (T2), and it is the fourth word in that document (position 3).

If we run a phrase search for "what is it" we get hits for all the three words in both documents 0 and 1. But the words occur only consecutively in document 1.

With the inverted index created, the query can now be resolved by jumping to the word id (via random access) in the inverted index. Random access is generally regarded as being faster than sequential access.

In Ai-Times, we build 3 inverted indexes to accelerate the random access with huge data.

2.2 MapReduce

Over the past few years, Google has published details of their infrastructure. Developers within Google are able to easily write algorithms that efficiently and reliably process many terabytes of data. To do this, they leverage two technologies: the Google File System (GFS), which implements reliable distributed storage; and MapReduce, a method of processing large data sets. Dean and Ghemawat [13] have written a paper to introduce MapReduce.

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key.

Programs written in this functional style are automatically executed in parallel on a large cluster of commodity machines. One can efficiently "grep" weeks of logs from a high-volume site, constructing concise summaries. One can build efficiently searchable indexes of huge datasets. Such tasks are easily implemented with little coding effort. Scalability and reliability are handled by the system.

In MapReduce, the computation takes a set of input key/value pairs, and produces a set of output key/value pairs. A whole round computation consists of two functions: Map and Reduce.

Map applies a function against each element of a list to get a transformed version of the list which consists of a set of intermediate key/value pairs. The intermediate key/value pairs will then be passed on to the Reduce. The Map function can be described as the following formula:

Map (k, v) -> <k', v'>.

where:

<k, v> is the input pairs which we called it "raw data".

<k', v'> is the intermediate pairs generated by Map function.

Reduce merges all intermediate values associated with the same intermediate key. It takes a function and runs it against items in the intermediate list, resulting in a single value for each invocation. The Reduce function can be described as following formula:

Reduce (k', v') -> <k', v''>.

where:

<k', v'> is the intermediate pairs generated by Map which is the input data to Reduce.

<k', v''> is the result pairs generated by Reduce. Reduce merges all intermediated values associated with the same intermediate key and obtains the derived value v'' associated with key k'.

2.2.1 A Simple Example

Here is a typical example in information retrieval system. Consider the problem of counting document frequency of each word in a large collection of documents. The problem can be interpreted by following Map and Reduce functions.

```
Map(String key, String value):
    // key: document name
    // value: document contents
    for each word w in value:
        EmitIntermediate(w, "1");
```

```
reduce(String key, Iterator values):
    // key: a word
    // values: a list of counts
    int result = 0;
        for each v in values:
            result += ParseInt(v);
        Emit(AsString(result));
```

Given the texts T0 = "it is what it is", T1 = "what is it" and T2 = "it is a banana", the Map function gets the following intermediate pairs:

<it, 1>, <is, 1>, <what, 1>, <it, 1>, <is, 1>

<what, 1>, <is, 1>, <it, 1>

<it, 1>, <is, 1>, <a, 1>, < banana, 1>

Map function returns each word with associated occurrences (say 1 here). The intermediate pairs are then passed on to the Reduce function. The result of the Reduce function is:

<it, 4>, <is, 4>, <what, 2>, <a, 1>, < banana, 1>

The Reduce function sums up all the occurrences for each particular word and returns the result.

2.2.2 A Typical Example in WNRS: Indexing

Indexing is an important module in an information retrieval system. Indexing module takes the extracted content as inputted data, segments the content individual words and then returns the <word, position_info1; position_info2...position_infok> list, and then write the result to the index.

The Map and Reduce functions are as follows:

```
Map(String url, String pure_content):
    // url: the url of the web page.
    // pure_content: The content after parsing and extraction.
   for(each word v in pure_content)
       EmitIntermediate(v, position_info);
    //position_info: The position information related to the word v.

Reduce(String word, Iterator position_infos):
    // word: a word
    // position_infos: a list of position_info
```

```
for each v in position_infos:
   position_info_list= position_info_list+”;”+ v
Emit(position_info_list);
```

The Map function segments the pure_content into word by word then returns the <word, position_info> pairs as intermediate result. The Reduce function merges the intermediate pairs and generates the <word, position_info1; position_info2...position_infok> as the output result. Using the same example as 2.2.1, the Map function gets the following intermediate pairs:

<it, (1,1)>, <is, (1,2)>, <what, (1,3)>, <it, (1,4)>, <is, (1,5)>

<what, (2,1)>, <is, (2,2)>, <it, (2,3)>

<it, (3,1)>, <is, (3,2)>, <a, (3,3)>, < banana, (3,4)>

The Reduce function merges the intermediate pairs and generates the following result:

<it, (1,1); (1,4); (2,3); (3,1); >.

<is, (1,2); (1,5); (2,2); (3,2>.

<what, (1,3); (2,1)>.

<a, (3,3)>.

< banana, (3,4)>

The intermediate pairs having same key value are merged into one pair by Reduce function, for example, <what, (1,3)> and <what, (2,1)> have the same key value “what”, as a result these two intermediate pairs are merged into <what, (1,3); (2,1)>.

2.3 Vector Space Model

The vector space model [15, 27] has been widely used in the traditional IR field. Most search engines also use similarity measures based on this model to rank Web documents. The model creates a space in which both documents and queries are represented by vectors. For a fixed collection of documents, an m-dimensional vector is generated for each document and each query forms sets of terms with associated weights, where m is the number of unique terms in the document collection. Then, a vector similarity function, such as the inner product, can be used to compute the similarity between a document and a query.

The classic vector space model as proposed by Salton et al. [27] had both local and global parameters incorporated in the term weight (w(ij)) equation [27] (known as the tf-idf [15]):

w(ij) = f(ij) x Log (D / d(j))

where:

w(ij) is the term weight for keyword term j in document i.

f(ij) is the frequency in which the term j occurred in the document i.

d(j) is the number of documents containing the term j, and,

D is the total number of documents.

Note that the quotient, d(ij)/D, is essentially the probability of finding the document containing the term n, in the document set being used and represents the global parameter.

The Vector Space Model has the following limitations:

- Long documents are considered poor representatives of the Vector Space Model because they had poor similarity values (a small scalar product and a large dimensionality).
- Documents with similar context but different term vocabulary ("False negative match", error of rejecting the documents which are actually relevant to the query).

- The search keywords were being typed during the search in an inappropriate manner giving poorer results e.g. key + ing, para + meter ("False positive match", error of not rejecting the documents which are not relevant to the query).

We implemented an improved VSM (vector space model) with query expansion in Ai-Times.

Chapter 3

Architecture and Algorithm of Ai-Times

Figure 1 illustrates the architecture of Ai-Times, which is made up of some typical web based IR system modules: a web crawler, an extraction module, an index engine, and a search module etc... In the following we describe our modifications, the proposed optimized web crawler module, the news extraction module, the summarization module and their algorithms.

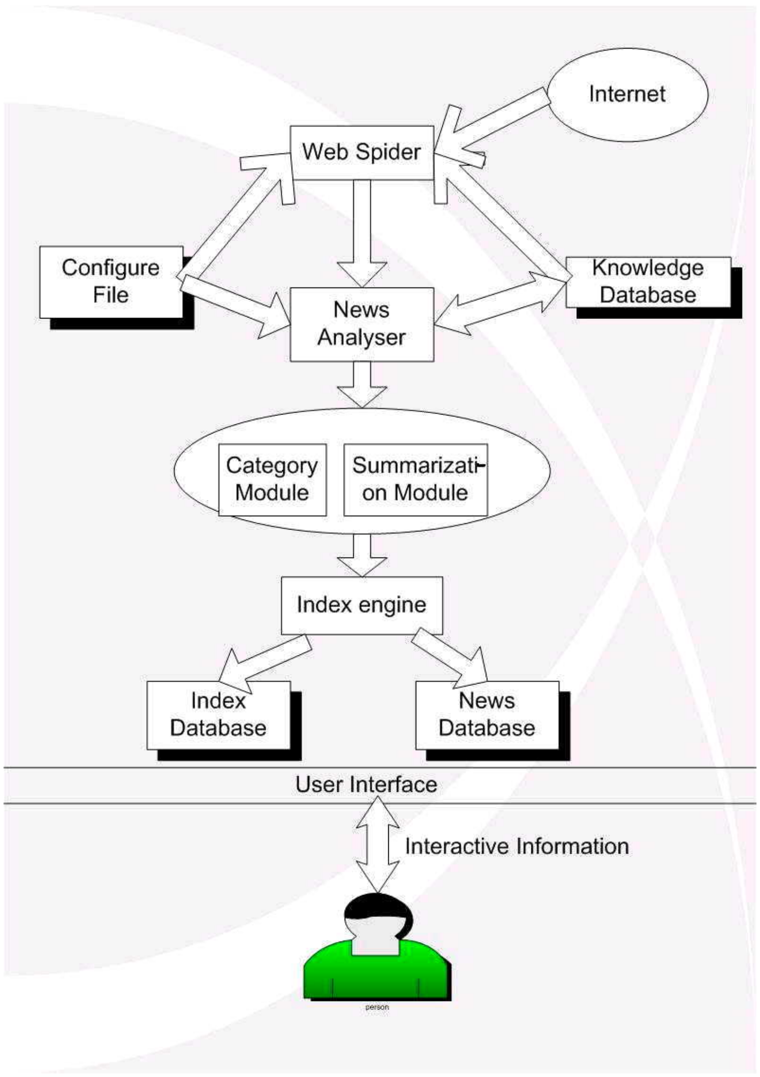


Fig. 1. The architecture of Ai-Times

3.1 Optimized Crawler Algorithm

One of the fundamental and important components of news information retrieval system is the Web Crawler that can collect the web document automatically. Much research has been done in this area, for instance, the Cobweb [30], is a typical Web Crawler. The classic crawler algorithm has been discussed in literature [11, 37]. These approaches [11, 30, 37], however, need to revisit all the web documents to update the news. To shorten the crawler's refresh time, the Ai-Times web crawler module is optimized for collecting news from predefined news web sites. The crawler automatically categorizes documents at these sites as being of three types: valueless web documents, news content web documents, or index/list web documents.

The definitions of valueless web document, news content web document, index/ list web document are given below:

Valueless web document

Valueless web documents are web documents that contribute nothing to the news retrieval, for example, an advertisement page, are not the object of news retrieval.

News content web document

News content web documents are web news documents that mainly contain the news text content or the news images or other multimedia sources.

Index/list web document

Index/list web documents are web documents that mainly contain hyperlinks linking to news content web documents with a relevant caption, which is often the news headline. This is the most frequently updated of these three types of web news document.

More than 90% of web documents of a news web site are news content web documents and they are seldom or never modified or updated. Typically, a crawler determines whether it is necessary to refresh a downloaded web document by sending a request to the web server and analyzing the return http header from the web server to get the last time the web document was modified. This not only wastes time and system resources, many http servers do not provide this information in the http header. One common refresh policy in this case is to revisit all web documents to find the update information. This is very time-consuming. An alternative refresh policy is to select and revisit important pages, but this method often leads to information loss [22].

The proposed modified web crawler can avoid refreshing irrelevant documents because its definition of the three types of web documents allows the web crawler to update only index/list web documents, thereby shortening the refresh interval.

The optimized crawler algorithm is as follows :

Algorithm I: Optimized Crawler Algorithm:

```
Begin
Let I be a list of initial URLs of the news website;
Let F be a queue;
  For each URL i in I
    Enqueue(i,F);
  End
  While F is not empty
    u=Dequeue(F);
    if u has not been processed
      Get (u);
      Case u’s type:
      Valueless web document:
        Skip u.
      News content web document:
        Store u;
      Index or list page:
        Extract the hyperlinks and relevant caption;
        Let U be the set of hyperlinks extracted;
```

```
            For each hyperlink u in U
               Enqueue(u,F);
            End
        Else—u has already been processed
           Case u’s type
              Valueless web document:
                 Skip u;
              News content web document:
                 Skip u;
              Index or list web document:
                 Update checking
        End
        End
End
```

It is often a time-consuming job to check whether a web document has been modified or updated. In general, the crawler should often rescan all the websites and all web documents for update checking.

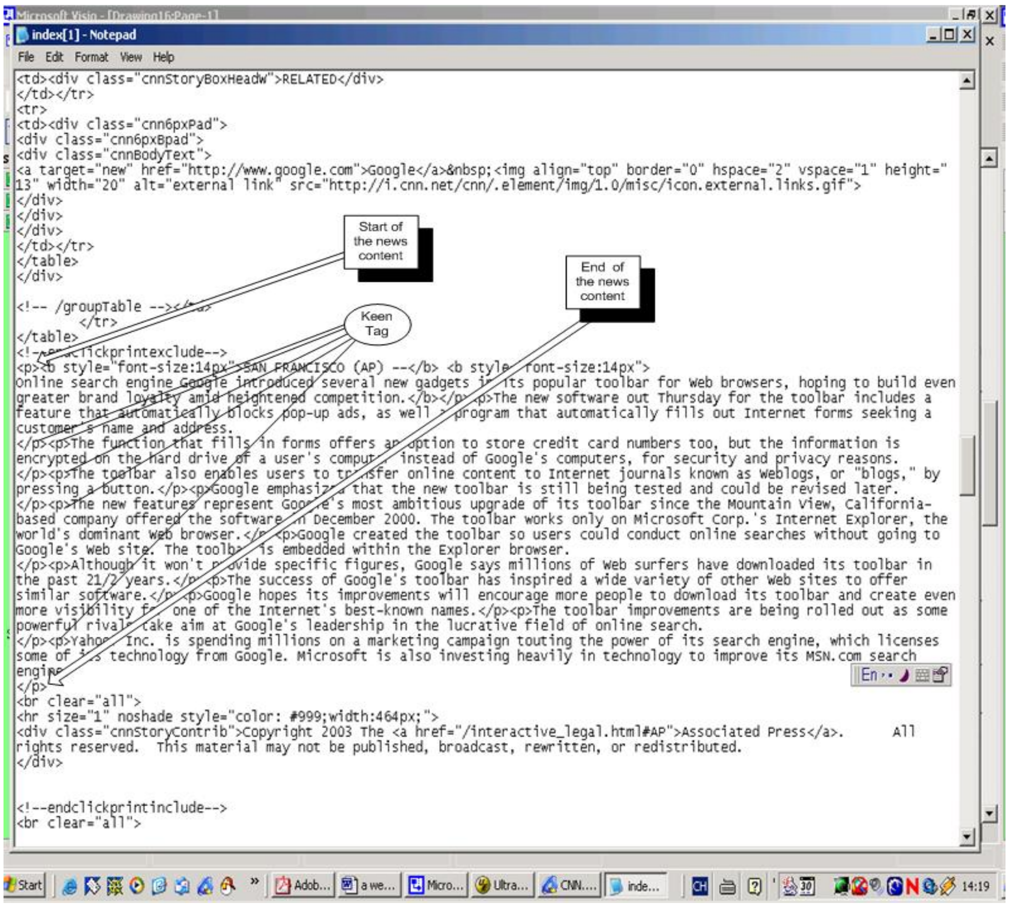
By categorize the web documents into 3 types, the Ai-Times web crawler needs not request all web documents during the update interval. As we can see from the Algorithm I, the optimized crawler will ignore the valueless web documents and news content web documents and only revisit the index/list web documents for update checking. This saves much time and makes the refresh interval shorter.

3.2 Keen-Tag based News Extraction Algorithm

To deal with the problem of extracting desired data from semi-structured documents, the Ai-Times news extraction module makes use of a keen tag extraction algorithm. Keen tags are defined as those tags that are most typically found in a webpage inside or around the news text content. These tags are used to help the news extraction module identify data rich parts of a web news document. Our research has shown that in most news web documents, the news text content string tokens always spatially cluster together in the html source code. The tags “<p>”,” </p>”,” <br>”,”

<font>”,” <img..>” will always be keen tags but other tags which are highly likely to signal data rich content are <font*>, </font>, <b>, <a href*>, </a>, <img*>, </br>, <strong>, </strong>, <div>, </div>, <center>, </center>.

Figure 2 shows an example of “keen tags” from a document at http://www.cnn.com.



Microsoft Visio - [Drawing16:Page-1]
index[1] - Notepad
File Edit Format View Help
Start of the news content
End of the news content
Keen Tag

```
<td><div class="cnnStoryBoxHeadw">RELATED</div>
</td></tr>
<tr>
<td><div class="cnn6pxPad">
<div class="cnn6pxBpad">
<div class="cnnBodyText">
<a target="new" href="http://www.google.com">Google</a>&nbsp;<img align="top" border="0" hspace="2" vspace="1" height="
13" width="20" alt="external link" src="http://i.cnn.net/cnn/.element/img/1.0/misc/icon.external.links.gif">
</div>
</div>
</div>
</td></tr>
</table>
</div>

<!-- /groupTable --></td>
        </tr>
</table>
<!--endclickprintexclude-->
<p><b style="font-size:14px">SAN FRANCISCO (AP) --</b> <b style="font-size:14px">
Online search engine Google introduced several new gadgets in its popular toolbar for Web browsers, hoping to build even
greater brand loyalty amid heightened competition.</b></p><p>The new software out Thursday for the toolbar includes a
feature that automatically blocks pop-up ads, as well a program that automatically fills out Internet forms seeking a
customer's name and address.
</p><p>The function that fills in forms offers an option to store credit card numbers too, but the information is
encrypted on the hard drive of a user's computer instead of Google's computers, for security and privacy reasons.
</p><p>The toolbar also enables users to transfer online content to Internet journals known as Weblogs, or "blogs," by
pressing a button.</p><p>Google emphasized that the new toolbar is still being tested and could be revised later.
</p><p>The new features represent Google's most ambitious upgrade of its toolbar since the Mountain View, California-
based company offered the software in December 2000. The toolbar works only on Microsoft Corp.'s Internet Explorer, the
world's dominant Web browser.</p><p>Google created the toolbar so users could conduct online searches without going to
Google's Web site. The toolbar is embedded within the Explorer browser.
</p><p>Although it won't provide specific figures, Google says millions of Web surfers have downloaded its toolbar in
the past 21/2 years.</p><p>The success of Google's toolbar has inspired a wide variety of other Web sites to offer
similar software.</p><p>Google hopes its improvements will encourage more people to download its toolbar and create even
more visibility for one of the Internet's best-known names.</p><p>The toolbar improvements are being rolled out as some
powerful rivals take aim at Google's leadership in the lucrative field of online search.
</p><p>Yahoo Inc. is spending millions on a marketing campaign touting the power of its search engine, which licenses
some of its technology from Google. Microsoft is also investing heavily in technology to improve its MSN.com search
engine.
</p>
<br clear="all">
<hr size="1" noshade style="color: #999;width:464px;">
<div class="cnnStoryContrib">Copyright 2003 The <a href="/interactive_legal.html#AP">Associated Press</a>.      All
rights reserved.  This material may not be published, broadcast, rewritten, or redistributed.
</div>


<!--endclickprintinclude-->
<br clear="all">
```

Fig. 2. A source example of the news content web document.

It can be seen that the news content strings clustered in the html source are accompanied with the tag “<p>” and “</p>”. Defining keen tags allows us to divide the html source code of the web document – which consists of html tags and string tokens - into three categories:

1. html tags that are keen tags.

2. html tags that are not keen tags, such as <!==* and <form>.

3. string tokens that are not html tags.

Algorithm II: News extraction algorithm

This algorithm first determines whether a document is a news content document and then extracts the data rich part from this document.

```
1.  Separate the html source code into many parts by the type 2 tags; therefore, each part contains the type1 and type3 strings. After that, the html source of the news document would look like this:
tag1-tag1-tag3-tag3-tag3-tag1  tag2 tag3-tag3-tag1  tag2 tag3-tag1

            part1                          part2              part3
2.  Score each part, the scoring formulation is:
Score (i)= $\sum_{k=1}^{ni} len(k)$                                     ----1
where:
i: the sequence of the part. For i=1,2,…n (n is the total number of the part).
ni: the total number of type3 tags in the current part.
k: the sequence of the type3 tags in the current part. For k=1,2,…ni.
len(k)—the length of the kth type3 tag in the current part.
3.   Select the score winner part as the data-rich part.
4.   Evaluate the selected part to see if it is a news content web document. The equation as below:
          evaluate()=fin(score)* tin(hrenum)             ----2
where:
score: the score obtained from equation 1.
hrefnum: the total number of hyperlink html tags(html tag like<a href…>) in that part.
                fin(k)=1 while k>N
                fin(k)=0 while $k \le N$
where:
N: the predefined low range of the news content's length. Any news document whose content length is shorter than N will not be considered to be a useful news web document.
                tin(k)=1 while k<T
                tin(k)=0 while $k \ge T$
where
T: the predefined up range of the total number of the link tags in a news content. Any news document that contains more than T link tags within the news content will not be considered as a useful news web document.
5. If the evaluate() = 1, this web document can be regarded as a news content web document, and then the selected data rich part will be extracted as the news text content.
```

We define a list containing universal keen tags as below:

The list: <p>, </p>, <font*>, </font>, <b>, <a href*>, </a>, <img*>, <br>, <strong>, </strong>, </div>, <div>, <center>, </center>.

Besides the universal keen tags we have defined before, there are still some website-related keen tags that we can derive from the training phrase. At the training phrase, the well-defined web news document will be inputted and the system will analyze these web news documents based on some pre-defined rules, then output the website-relevant keen tags.

3.3 Automatic Summarization Algorithm

The amount of news information available electronically has grown dramatically. There is an increasing demand for domain-independent summarization method. Ai-Times can generate the news summary automatically. In this section we describe the automatic summarization module of Ai-Times. The process of Ai-Times automatic summarization consists of the following three steps: characteristic words generation, sentence weighting and summary generation. In section 3.4.1 we describe the characteristic words generation. In section 3.4.2 we describe the sentence weighting and in section 3.4.3 we describe the summary generation.

3.3.1 Characteristic Words Generation

Characteristic words are generated by word segmentation algorithm and characteristic word weighting equation based on Statistical and Probabilistic Approach [4]. Words in web news document are assigned to different weighting based on their importance and frequency in document. Words with highest weighting are extracted as characteristic words. The occurrence frequency of a word in a document and the number of documents containing it are important factors to calculating the characteristic weight.

The summarization algorithm scans a web news document and finds out proper words in the web news document based on the word occurrence frequency and word length, then, uses the following equation:

$$P(w) = F_t(w) \bullet \left(1 - \frac{numdoc}{tota \ln umdoc}\right) \bullet \left(L(w) - D\right)^c$$

where:

w is the Chinese word extracted from a sentence in the web news document

P(w) is the weight of w

Ft(w) is the frequency of w

L(w) is the length of w

numdoc is the number of documents which contains w

totalnumdoc is the total number of documents

D is the minimum length of w

c is the influence of the length of the sequence of Chinese characters

Characteristic words can only be treated as keywords for retrieving the relevant web news documents. These words carry no semantic meaning.

3.3.2 Sentence Weighting

So far, we have described how to generate characteristic words. Note that the second step of our summarization algorithm is sentence weighting. In this step sentences are assigned weights, which indicate their importance in a web news document. The higher the weight, the greater the importance. The highest weight is assigned to the sentence that presents the most important concept of the web news document. Important sentences must have the features of more characteristic words in the sentence; of higher characteristic weight of characteristic words in the sentence; of shorter length of sentences; of having less sub-sentences for an important sentence;

and of less numeral words in the sentence. The following equation computes the sentence’s importance weight.

$$L(s) = \frac{\sum_{i=1}^{N} L_i}{s_0 \bullet s_1 \bullet s_2 \bullet m}$$

where:

N is the number of Characteristic words in the sentence.

L_i , i=1 to N, is characteristic weight of the ith characteristic word in the sentence.

s_0 is the total number of words in the sentence.

s_1 is the number of sub-sentences in the sentence.

s_2 is the number of numeral words in the sentence.

m is an integer variable. Normally, it is set to 1.

The sentence containing more characteristic words will be assigned a higher weight. The sentence containing more number of sub-sentences, words and numeral will be assigned a smaller weight. The title and sub-title of the document and the main sentence in the paragraph would be assigned a higher sentence weight dynamically. Important sentences can then be extracted from a document.

3.3.3 Summary Generation

The third step of Ai-Times summarization algorithm is summary generation. In this step, the sentences with the highest sentence weight are extracted to generate a summary in a particular ratio required by the user according to the following equation. The created summary is the compressed version of the original document. Different summary ratios can create different quality of the summary.

$$Abstract = \sum_{i=1}^{x} | s_i | = L \bullet R$$

where:

L is the length of the web news document.

R is the abstract ratio.

s_i is the sentence with ith highest sentence weight.

$|s_i|$ is the length of s_i.

3.4 Experiments

In this chapter we describe experiments and results on the comparison of the update speed of the Ai-Times single optimized crawler with the single traditional crawler; we evaluate the news extraction module. In each case Ai-times is run on several PC servers with the following configuration: CPU: p4 2.4 GHz; RAM: 512 MB; Bandwidth: 1MB.

3.4.1 Performance of Optimized Crawler

In this experiment we compare and test the speed with which the optimized crawler updates downloaded documents, the update interval. Though it is difficult for us to obtain the exact consumed time of the update interval of the crawler, we measured the update interval as follows. We can compare our optimized crawler algorithm with traditional crawler algorithm in the following way:

In the experiment, our crawler has processed 3.53 million web documents in roughly 6 days. Assume that the traditional crawler has also processed the same number of web documents in a certain days; we then start a new round of web crawling. As stated earlier, our optimized crawler should only revisit the index/list web documents (in our experiment the number of the index /list web documents is

0.37 million) while the traditional crawler should revisit all web documents (in our experiment the number of all web documents is 3.53 millions). The consumed time is linear to the number of the web documents the crawler visits.

Figure 3 shows the improvement in terms of the performance of our optimized crawler algorithm compared to the traditional algorithms. The traditional crawlers revisited 3.53 millions and 2.47 millions web documents respectively, while our proposed optimized crawler only revisited 0.37 millions web documents. The update interval of the optimized crawler is substantially 85% shorter than the traditional crawlers.

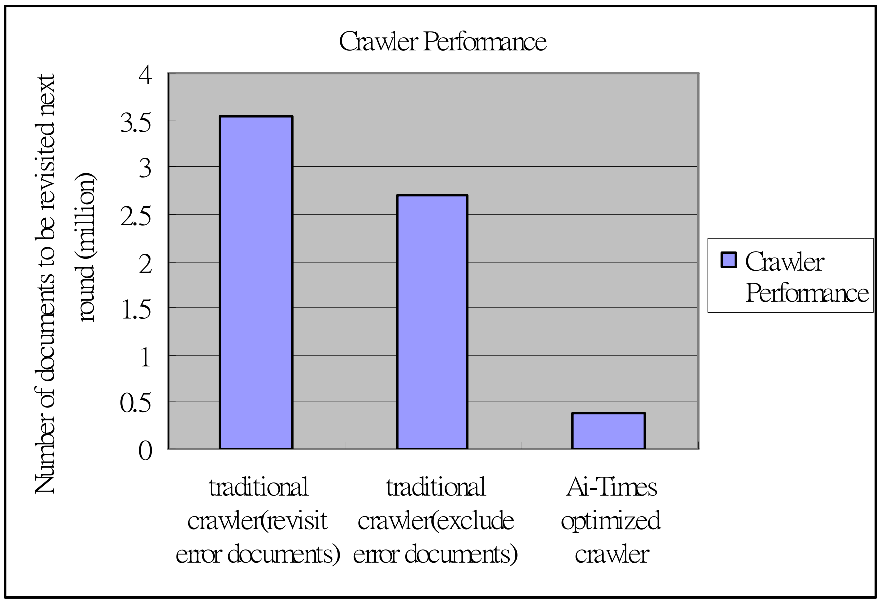


Fig. 3. Compare the optimized crawler with the traditional crawler.

3.4.2 News Extraction

In previous experiment, we defined a keen tag list for global use:

<p *>, </p>, <font *>, </font>, <b>, <a *>, </a>, <img *>, <br *>,</br>, <strong>, </strong>, </div>, <div *>, <center>, </center>. By using this keen tag list, our system has downloaded totally 1.46 million news documents.

To measure the effectiveness of the news extraction module, two ratios are used: precision and recall. Precision is the ratio of the number of correctly fetched web news documents to the total number of fetched web news documents that may contain some rubbish. Recall is the ratio of number of correctly fetched web news documents to the total number of web news documents in all predefined news websites. Because it is difficult for us to obtain the total number of web news documents or to verify each fetched web document to see whether it is correct or wrong, we approximate the recall and precision in the following way:

$$pre=num(r1)/num(N1) \tag{1}$$

where:

pre is the precision.

N1 is a set of documents randomly selected from fetched web news documents.

r1 is the correctly fetched web news documents in N1.

num(S) is the function to obtain the number of the units in collection S.

$$rec=num(r2)/num(N2) \tag{2}$$

where:

rec is the recall.

N2 is a set of documents randomly selected from total web news documents.

r2 is the web news documents in N2 fetched by our system.

num(S) is also the function to obtain the number of the units in collection S.

We then select 1,000 web news documents from fetched web news documents and the number of correctly fetched documents in them is 915. We also select 1,000 web news documents from total web news documents and the number of fetched web news documents is 876. Therefore, we can obtain the approximation as following:

$$pre=915/1000=0.915$$

$$rec=876/1000=0.876$$

Note that we use the “keen tag” analyzing method to extract the news content so we need not implement different wrappers for different news websites; the effectiveness of our system is relatively acceptable.

Our system also provides a friendly UI for the user to access the news archive. It can be shown as below:

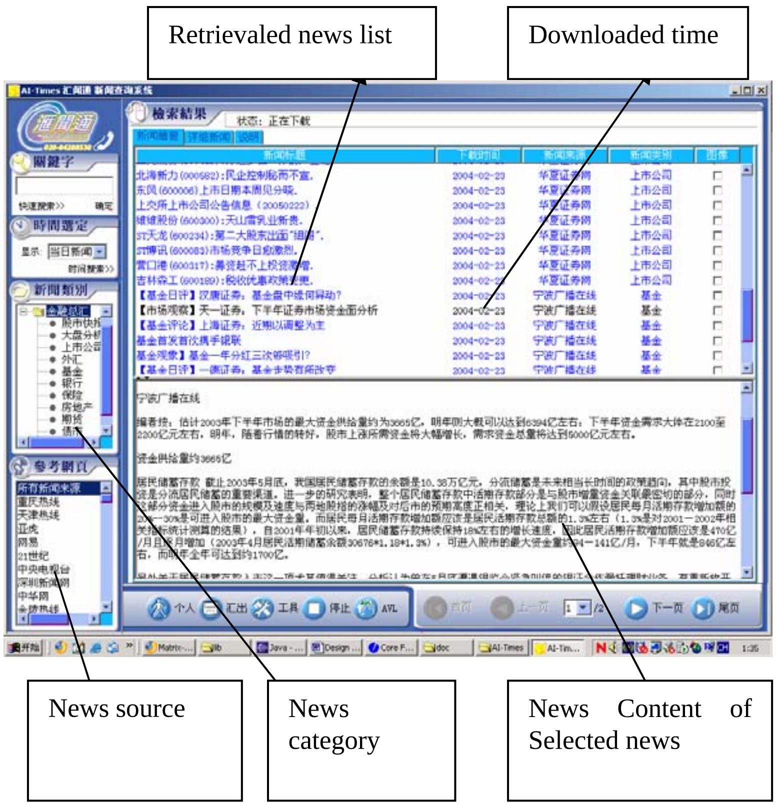
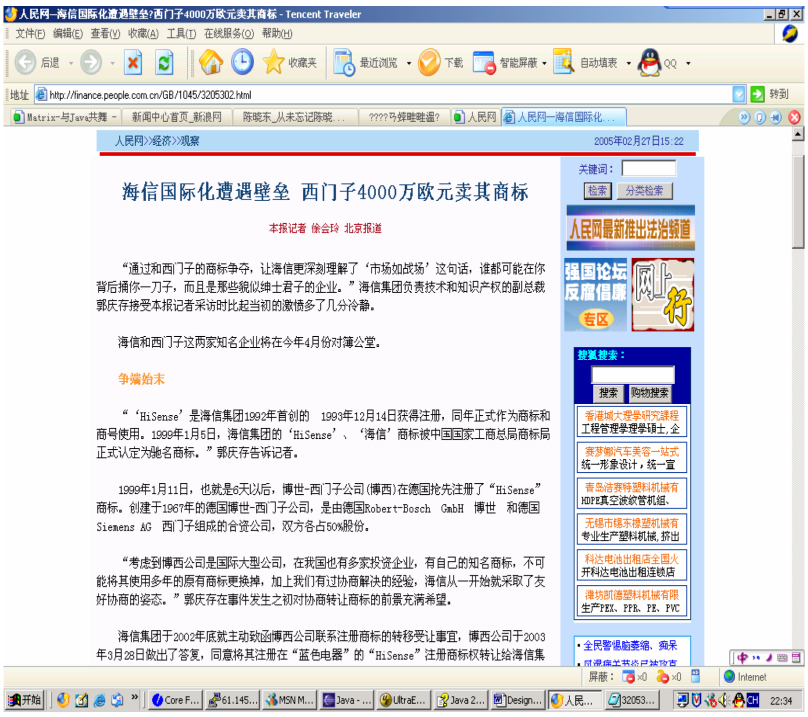


Fig. 4. User interface of Ai-times

Below is an example of extracting process:



http://finance.people.com.cn/GB/1045/3205302.html

人民网>>经济>>观察 2005年02月27日15:22

海信国际化遭遇壁垒 西门子4000万欧元卖其商标

本报记者 徐会玲 北京报道

“通过和西门子的商标争夺，让海信更深刻理解了‘市场如战场’这句话，谁都可能在你背后捅你一刀子，而且是那些貌似绅士君子的企业。”海信集团负责技术和知识产权的副总裁郭庆存接受本报记者采访时比起当初的激愤多了几分冷静。

海信和西门子这两家知名企业将在今年4月份对簿公堂。

争端始末

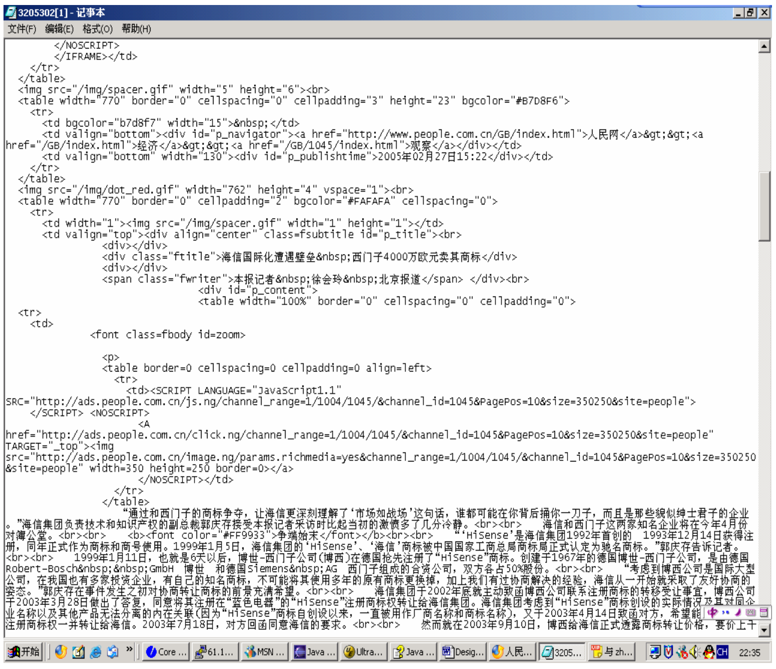
“‘HiSense’是海信集团1992年首创的 1993年12月14日获得注册，同年正式作为商标和商号使用。1999年1月5日，海信集团的‘HiSense’、‘海信’商标被中国国家工商总局商标局正式认定为驰名商标。”郭庆存告诉记者。

1999年1月11日，也就是6天以后，博世-西门子公司(博西)在德国抢先注册了“HiSense”商标。创建于1967年的德国博世-西门子公司，是由德国Robert-Bosch GmbH 博世 和德国Siemens AG 西门子组成的合资公司，双方各占50%股份。

“考虑到博西公司是国际大型公司，在我国也有多家投资企业，有自己的知名商标，不可能将其使用多年的原有商标更换掉，加上我们有过协商解决的经验，海信从一开始就采取了友好协商的姿态。”郭庆存在事件发生之初对协商转让商标的前景充满希望。

海信集团于2002年底就主动致函博西公司联系注册商标的转移受让事宜，博西公司于2003年3月28日做出了答复，同意将其注册在“蓝色电器”的“HiSense”注册商标权转让给海信集

Fig. 5. The web news document



```
        </NOSCRIPT>
        </IFRAME></td>
    </tr>
  </table>
  <img src="/img/spacer.gif" width="5" height="6"><br>
  <table width="770" border="0" cellspacing="0" cellpadding="3" height="23" bgcolor="#B7D8F6">
    <tr>
      <td bgcolor="b7d8f7" width="15">&nbsp;</td>
      <td valign="bottom"><div id="p_navigator"><a href="http://www.people.com.cn/GB/index.html">人民网</a>&gt;&gt;<a
href="/GB/index.html">经济</a>&gt;&gt;<a href="/GB/1045/index.html">观察</a></div></td>
      <td valign="bottom" width="130"><div id="p_publishtime">2005年02月27日15:22</div></td>
    </tr>
  </table>
  <img src="/img/dot_red.gif" width="762" height="4" vspace="1"><br>
  <table width="770" border="0" cellpadding="2" bgcolor="#FAFAFA" cellspacing="0">
    <tr>
      <td width="1"><img src="/img/spacer.gif" width="1" height="1"></td>
      <td valign="top"><div align="center" class=fsubtitle id="p_title"><br>
                <div></div>
                <div class="ftitle">海信国际化遭遇壁垒&nbsp;西门子4000万欧元卖其商标</div>
                <div></div>
                <span class="fwriter">本报记者&nbsp;徐会玲&nbsp;北京报道</span> </div><br>
                                <div id="p_content">
                                <table width="100%" border="0" cellspacing="0" cellpadding="0">
  <tr>
    <td>
              <font class=fbody id=zoom>

                <p>
                <table border=0 cellspacing=0 cellpadding=0 align=left>
                  <tr>
                    <td><SCRIPT LANGUAGE="JavaScript1.1"
SRC="http://ads.people.com.cn/js.ng/channel_range=1/1004/1045/&channel_id=1045&PagePos=10&size=350250&site=people">
    </SCRIPT> <NOSCRIPT>
                          <A
href="http://ads.people.com.cn/click.ng/channel_range=1/1004/1045/&channel_id=1045&PagePos=10&size=350250&site=people"
TARGET="_top"><img
src="http://ads.people.com.cn/image.ng/params.richmedia=yes&channel_range=1/1004/1045/&channel_id=1045&PagePos=10&size=350250
&site=people" width=350 height=250 border=0></a>
                          </NOSCRIPT></td>
                    </tr>
                </table>
                    “通过和西门子的商标争夺，让海信更深刻理解了‘市场如战场’这句话，谁都可能在你背后捅你一刀子，而且是那些貌似绅士君子的企业
。”海信集团负责技术和知识产权的副总裁郭庆存接受本报记者采访时比起当初的激愤多了几分冷静。<br><br>    海信和西门子这两家知名企业将在今年4月份
对簿公堂。<br><br>    <b><font color="#FF9933">争端始末</font></b><br><br>    “‘HiSense’是海信集团1992年首创的  1993年12月14日获得注
册，同年正式作为商标和商号使用。1999年1月5日，海信集团的‘HiSense’、‘海信’商标被中国国家工商总局商标局正式认定为驰名商标。”郭庆存告诉记者。
<br><br>    1999年1月11日，也就是6天以后，博世-西门子公司(博西)在德国抢先注册了“HiSense”商标。创建于1967年的德国博世-西门子公司，是由德国
Robert-Bosch&nbsp;&nbsp;GmbH  博世  和德国Siemens&nbsp;AG  西门子组成的合资公司，双方各占50%股份。<br><br>    “考虑到博西公司是国际大型
公司，在我国也有多家投资企业，有自己的知名商标，不可能将其使用多年的原有商标更换掉，加上我们有过协商解决的经验，海信从一开始就采取了友好协商的
姿态。”郭庆存在事件发生之初对协商转让商标的前景充满希望。<br><br>    海信集团于2002年底就主动致函博西公司联系注册商标的转移受让事宜，博西公司
于2003年3月28日做出了答复，同意将其注册在“蓝色电器”的“HiSense”注册商标权转让给海信集团。海信集团考虑到“HiSense”商标创设的实际情况及其对同企
业名称以及其他产品无法分离的内在关联(因为“HiSense”商标自创设以来，一直被用作厂商名称和商标名称)，又于2003年4月14日致函对方，希望能
注册商标权一并转让给海信。2003年7月18日，对方回函同意海信的要求。<br><br>    然而就在2003年9月10日，博西给海信正式透露商标转让价格，要价上千
```

Fig. 6. The html source code of the news

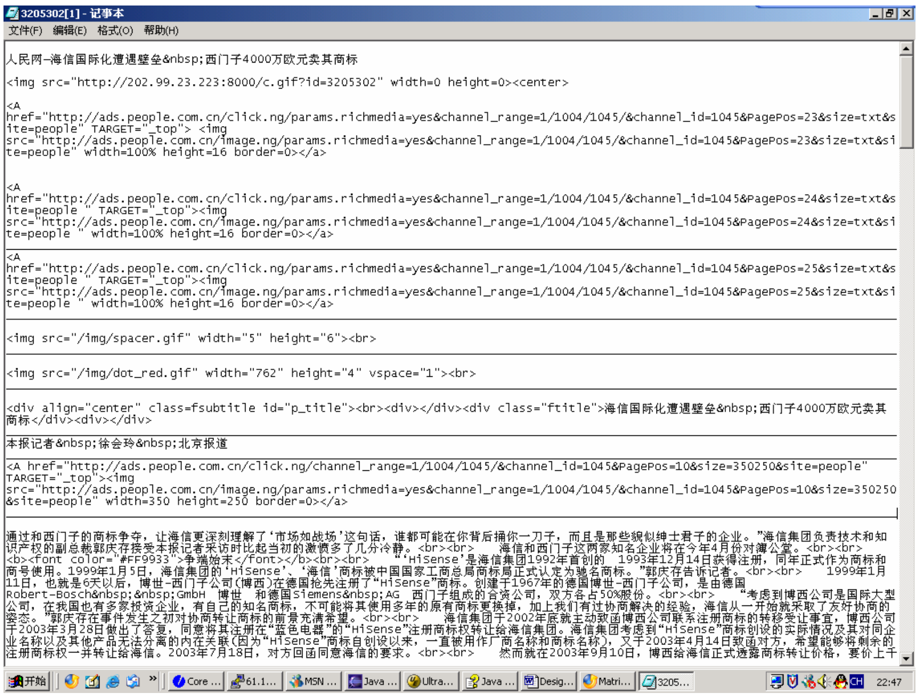
Separated the html source code into many parts. The result can be shown as figure 7.

From Figure 8 we can see that system has scored each part and selected the part having the highest score as the data rich part.

Evaluated the selected data rich part, N was defined to be 100 and T was defined to be 5, thus the evaluate function is as below:

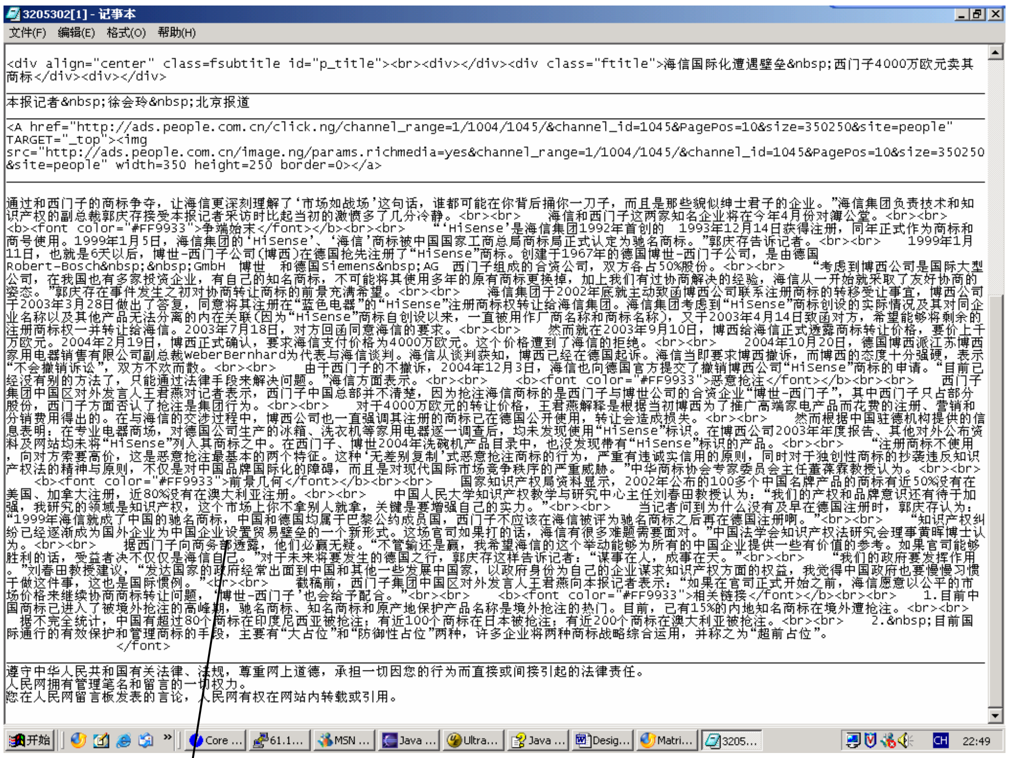
$$evaluate = fin\,(2159) \bullet tin\,(0) = 1 \bullet 1 = 1$$

As the length of the extracted data rich part is longer than the previously defined value 100 and the number of the hyperlinks is not more than 5. Thus the selected part was extracted to be the news content and the web document was classified as a news content web document.



```
人民网—海信国际化遭遇壁垒&nbsp;西门子4000万欧元卖其商标

<img src="http://202.99.23.223:8000/c.gif?id=3205302" width=0 height=0><center>

<A
href="http://ads.people.com.cn/click.ng/params.richmedia=yes&channel_range=1/1004/1045/&channel_id=1045&PagePos=23&size=txt&s
ite=people" TARGET="_top"> <img
src="http://ads.people.com.cn/image.ng/params.richmedia=yes&channel_range=1/1004/1045/&channel_id=1045&PagePos=23&size=txt&si
te=people" width=100% height=16 border=0></a>


<A
href="http://ads.people.com.cn/click.ng/params.richmedia=yes&channel_range=1/1004/1045/&channel_id=1045&PagePos=24&size=txt&s
ite=people " TARGET="_top"><img
src="http://ads.people.com.cn/image.ng/params.richmedia=yes&channel_range=1/1004/1045/&channel_id=1045&PagePos=24&size=txt&si
te=people " width=100% height=16 border=0></a>
________________________________________________________________
<A
href="http://ads.people.com.cn/click.ng/params.richmedia=yes&channel_range=1/1004/1045/&channel_id=1045&PagePos=25&size=txt&s
ite=people " TARGET="_top"><img
src="http://ads.people.com.cn/image.ng/params.richmedia=yes&channel_range=1/1004/1045/&channel_id=1045&PagePos=25&size=txt&si
te=people " width=100% height=16 border=0></a>
________________________________________________________________

<img src="/img/spacer.gif" width="5" height="6"><br>
________________________________________________________________

<img src="/img/dot_red.gif" width="762" height="4" vspace="1"><br>
________________________________________________________________

<div align="center" class=fsubtitle id="p_title"><br><div></div><div class="ftitle">海信国际化遭遇壁垒&nbsp;西门子4000万欧元卖其
商标</div><div></div>
________________________________________________________________
本报记者&nbsp;徐会玲&nbsp;北京报道
________________________________________________________________
<A href="http://ads.people.com.cn/click.ng/channel_range=1/1004/1045/&channel_id=1045&PagePos=10&size=350250&site=people"
TARGET="_top"><img
src="http://ads.people.com.cn/image.ng/params.richmedia=yes&channel_range=1/1004/1045/&channel_id=1045&PagePos=10&size=350250
&site=people" width=350 height=250 border=0></a>
________________________________________________________________

通过和西门子的商标争夺，让海信更深刻理解了‘市场如战场’这句话，谁都可能在你背后捅你一刀子，而且是那些貌似绅士君子的企业。”海信集团负责技术和知
识产权的副总裁郭庆存接受本报记者采访时比起当初的激愤多了几分冷静。<br><br>    海信和西门子这两家知名企业将在今年4月份对簿公堂。<br><br>
<b><font color="#FF9933">争端始末</font></b><br><br>    “‘HiSense’是海信集团1992年首创的  1993年12月14日获得注册，同年正式作为商标和
商号使用。1999年1月5日，海信集团的‘HiSense’、‘海信’商标被中国国家工商总局商标局正式认定为驰名商标。”郭庆存告诉记者。<br><br>    1999年1月
11日，也就是6天以后，博世-西门子公司(博西)在德国抢先注册了“HiSense”商标。创建于1967年的德国博世-西门子公司，是由德国
Robert-Bosch&nbsp;&nbsp;GmbH  博世  和德国Siemens&nbsp;AG  西门子组成的合资公司，双方各占50%股份。<br><br>    “考虑到博西公司是国际大型
公司，在我国也有多家投资企业，有自己的知名商标，不可能将其使用多年的原有商标更换掉，加上我们有过协商解决的经验，海信从一开始就采取了友好协商的
姿态。”郭庆存在事件发生之初对协商转让商标的前景充满希望。<br><br>    海信集团于2002年底就主动致函博西公司联系注册商标的转移受让事宜，博西公司
于2003年3月28日做出了答复，同意将其注册在“蓝色电器”的“HiSense”注册商标权转让给海信集团。海信集团考虑到“HiSense”商标创设的实际情况及其对同企
业名称以及其他产品无法分离的内在关联(因为“HiSense”商标自创设以来，一直被用作厂商名称和商标名称)，又于2003年4月14日致函对方，希望能够将剩余的
注册商标权一并转让给海信。2003年7月18日，对方回函同意海信的要求。<br><br>    然而就在2003年9月10日，博西给海信正式透露商标转让价格，要价上千
```

Fig. 7. Separated by type2 tag



```
<div align="center" class=fsubtitle id="p_title"><br><div></div><div class="ftitle">海信国际化遭遇壁垒&nbsp;西门子4000万欧元卖其
商标</div><div></div>

本报记者&nbsp;徐会玲&nbsp;北京报道

<A href="http://ads.people.com.cn/click.ng/channel_range=1/1004/1045/&channel_id=1045&PagePos=10&size=350250&site=people"
TARGET="_top"><img
src="http://ads.people.com.cn/image.ng/params.richmedia=yes&channel_range=1/1004/1045/&channel_id=1045&PagePos=10&size=350250
&site=people" width=350 height=250 border=0></a>

通过和西门子的商标争夺，让海信更深刻理解了‘市场如战场’这句话，谁都可能在你背后捅你一刀子，而且是那些貌似绅士君子的企业。”海信集团负责技术和知
识产权的副总裁郭庆存接受本报记者采访时比起当初的激愤多了几分冷静。<br><br>    海信和西门子这两家知名企业将在今年4月份对簿公堂。<br><br>
<b><font color="#FF9933">争端始末</font></b><br><br>    “‘HiSense’是海信集团1992年首创的  1993年12月14日获得注册，同年正式作为商标和
商号使用。1999年1月5日，海信集团的‘HiSense’、‘海信’商标被中国国家工商总局商标局正式认定为驰名商标。”郭庆存告诉记者。<br><br>    1999年1月
11日，也就是6天以后，博世-西门子公司(博西)在德国抢先注册了“HiSense”商标。创建于1967年的德国博世-西门子公司，是由德国
Robert-Bosch&nbsp;&nbsp;GmbH  博世  和德国Siemens&nbsp;AG  西门子组成的合资公司，双方各占50%股份。<br><br>    “考虑到博西公司是国际大型
公司，在我国也有多家投资企业，有自己的知名商标，不可能将其使用多年的原有商标更换掉，加上我们有过协商解决的经验，海信从一开始就采取了友好协商的
姿态。”郭庆存在事件发生之初对协商转让商标的前景充满希望。<br><br>    海信集团于2002年底就主动致函博西公司联系注册商标的转移受让事宜，博西公司
于2003年3月28日做出了答复，同意将其注册在“蓝色电器”的“HiSense”注册商标权转让给海信集团。海信集团考虑到“HiSense”商标创设的实际情况及其对同企
业名称以及其他产品无法分离的内在关联(因为“HiSense”商标自创设以来，一直被用作厂商名称和商标名称)，又于2003年4月14日致函对方，希望能够将剩余的
注册商标权一并转让给海信。2003年7月18日，对方回函同意海信的要求。<br><br>    然而就在2003年9月10日，博西给海信正式透露商标转让价格，要价上千
万欧元。2004年2月19日，博西正式确认，要求海信支付价格为4000万欧元。这个价格遭到了海信的拒绝。<br><br>    2004年10月20日，德国博西派江苏博西
家用电器销售有限公司副总裁weberBernhard为代表与海信谈判。海信从谈判获知，博西已经在德国起诉。海信当即要求博西撤诉，而博西的态度十分强硬，表示
“不会撤销诉讼”，双方不欢而散。<br><br>    由于西门子的不撤诉，2004年12月3日，海信也向德国官方提交了撤销博西公司“HiSense”商标的申请。“目前已
经没有别的方法了，只能通过法律手段来解决问题。”海信方面表示。<br><br>    <b><font color="#FF9933">恶意抢注</font></b><br><br>    西门子
集团中国区对外发言人王君燕对记者表示，西门子中国总部并不清楚，因为抢注海信商标的是西门子与博世公司的合资企业“博世-西门子”，其中西门子只占部分
股份，西门子方面否认了抢注是集团行为。<br><br>    对于4000万欧元的转让价格，王君燕解释是根据当初博西为了推广高端家电产品而花费的注册、营销和
分销费用得出的。在与海信的交涉过程中，博西公司也一直强调其注册的商标已在德国公开使用，转让会造成损失。<br><br>    然而根据中国驻德机构提供的信
息表明：在专业电器商场，对德国公司生产的冰箱、洗衣机等家用电器逐一调查后，均未发现使用“HiSense”标识。在博西公司2003年年度报告、其他对外公布资
料及网站均未将“HiSense”列入其商标之中。在西门子、博世2004年洗碗机产品目录中，也没发现带有“HiSense”标识的产品。<br><br>    “注册商标不使用
，向对方索要高价，这是恶意抢注最基本的两个特征。这种‘无差别复制’式恶意抢注商标的行为，严重有违诚实信用的原则，同时对于独创性商标的抄袭违反知识
产权法的精神与原则，不仅是对中国品牌国际化的障碍，而且是对现代国际市场竞争秩序的严重威胁。”中华商标协会专家委员会主任董葆霖教授认为。<br><br>
    <b><font color="#FF9933">前景几何</font></b><br><br>    国家知识产权局资料显示，2002年公布的100多个中国名牌产品的商标有近50%没有在
美国、加拿大注册，近80%没有在澳大利亚注册。<br><br>    中国人民大学知识产权教学与研究中心主任刘春田教授认为：“我们的产权和品牌意识还有待于加
强，我研究的领域是知识产权，这个市场上你不拿别人就拿，关键是要增强自己的实力。”<br><br>    当记者问到为什么没有及早在德国注册时，郭庆存认为：
“1999年海信就成了中国的驰名商标，中国和德国均属于巴黎公约成员国，西门子不应该在海信被评为驰名商标之后再在德国注册啊。”<br><br>    “知识产权纠
纷已经逐渐成为国外企业为中国企业设置贸易壁垒的一个新形式。这场官司如果打的话，海信有很多难题需要面对。”中国法学会知识产权法研究会理事黄晖博士认
为。<br><br>    据西门子向商务部透露，他们必赢无疑。“不管输还是赢，我希望海信的这个举动能够为所有的中国企业提供一些有价值的参考。如果官司能够
胜利的话，受益者决不仅仅是海信自己。”对于未来将要发生的德国之行，郭庆存这样告诉记者：“谋事在人，成事在天。”<br><br>    “我们的政府要发挥作用
。”刘春田教授建议，“发达国家的政府经常出面到中国和其他一些发展中国家，以政府身份为自己的企业谋求知识产权方面的权益，我觉得中国政府也要慢慢习惯
于做这件事，这也是国际惯例。”<br><br>    截稿前，西门子集团中国区对外发言人王君燕向本报记者表示：“如果在官司正式开始之前，海信愿意以公平的市
场价格来继续协商商标转让问题，‘博世-西门子’也会给予配合。”<br><br>    <b><font color="#FF9933">相关链接</font></b><br><br>    1.目前中
国商标已进入了被境外抢注的高峰期，驰名商标、知名商标和原产地保护产品名称是境外抢注的热门。目前，已有15%的内地知名商标在境外遭抢注。<br><br>
  据不完全统计，中国有超过80个商标在印度尼西亚被抢注；有近100个商标在日本被抢注；有近200个商标在澳大利亚被抢注。<br><br>    2.&nbsp;目前国
际通行的有效保护和管理商标的手段，主要有“大占位”和“防御性占位”两种，许多企业将两种商标战略综合运用，并称之为“超前占位”。
                </font>

遵守中华人民共和国有关法律、法规，尊重网上道德，承担一切因您的行为而直接或间接引起的法律责任。
人民网拥有管理笔名和留言的一切权力。
您在人民网留言板发表的言论，人民网有权在网站内转载或引用。
```

Fig. 8. Select the data rich part

The scoring winner part

3.5 Conclusion

In this chapter we presented our modifications, the proposed optimized web crawler module, the news extraction module and the summarization module and their algorithms. By distinguishing the web document as important and unimportant, the optimized crawler can fetch the web news nearly 6 times faster than the traditional crawler during the update interval. Using the keen tag based extraction algorithm, the news extraction module can extract the news content from the web news document with nearly zero manual effort.

In next chapter, we will discuss another two important modules in Ai-Times: Indexing module and Retrieval module.

Chapter 4

Indexing and Retrieval Strategy

The headache of building an effective search engine is that most of the data, typically documents in words, are not structured in a common interface. Although it is impossible to get exact relevant information for each user because the final judgment depends on individual user, it is achievable to get fairly appropriate information which met the user's need. There exist numerous algorithms [15] for information retrieval. In Ai-Times, we try to grant the maximum relevance information by applying the concept of Vector Space Model, Term Reweighting, Query Expansion as well as Inverted Indexing structures. Figure 9 shows the index and retrieval architecture:

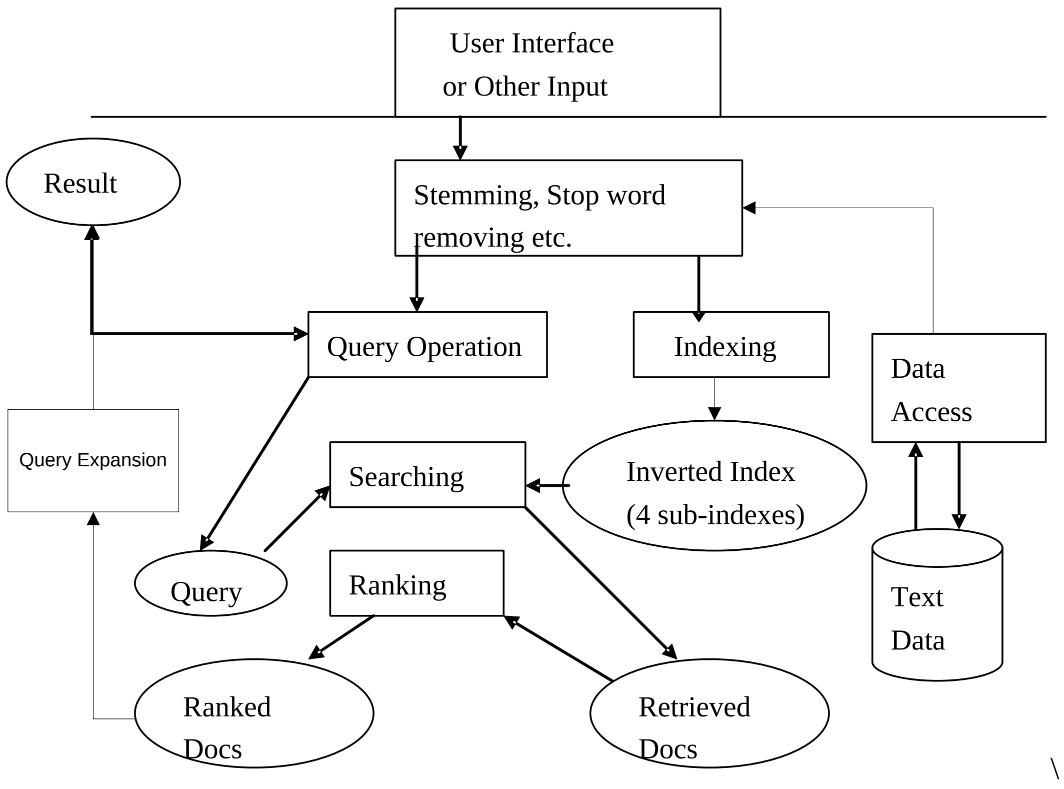


Fig. 9. Index and retrieval architecture

4.1 Inverted Indexing Strategy

In order to speed up the search, we build an inverted index over the data. Our inverted index's structure can be shown in Figure 10:

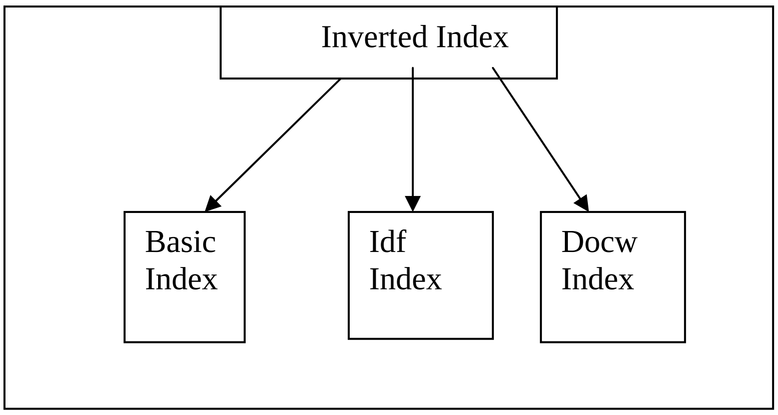


Fig. 10. Inverted index in Ai-Times

Inverted index in Ai-Times contains 3 sub-indexes; each sub-index is an inverted index format. The definition of the 3 sub-indexes is given as below:

Basic Index: the inverted index contains the [term-doc] information. The basic index allows us to retrieve the documents containing a specified term quickly.

Idf Index: the inverted index contains the [term-termidf] information. The idf index allows us to retrieve the idf value of a specified term quickly. The idf (inverse document frequency) is a measure of the general importance of the term:

$$Idf_j = Log(D / d_j)$$

where:

Idf_j is the inverse document frequency of the term j.

D is the number of the total documents.

d_j is the number of documents containing the term j.

Docw Index is the inverted index contains the [doc-docweight] information. The docw index allows us to retrieve the docw value of a specified document quickly.

$$docw_j = \sqrt{\sum_{k=1}^{n_j} (w_{kj})^2}$$

where:

$docw_j$ is the document weight of the document j;

w_{kj} is the idf value of the kth term in document j;

n_j is the number of the terms in document j;

Figure 11 shows the indexing execution flow:

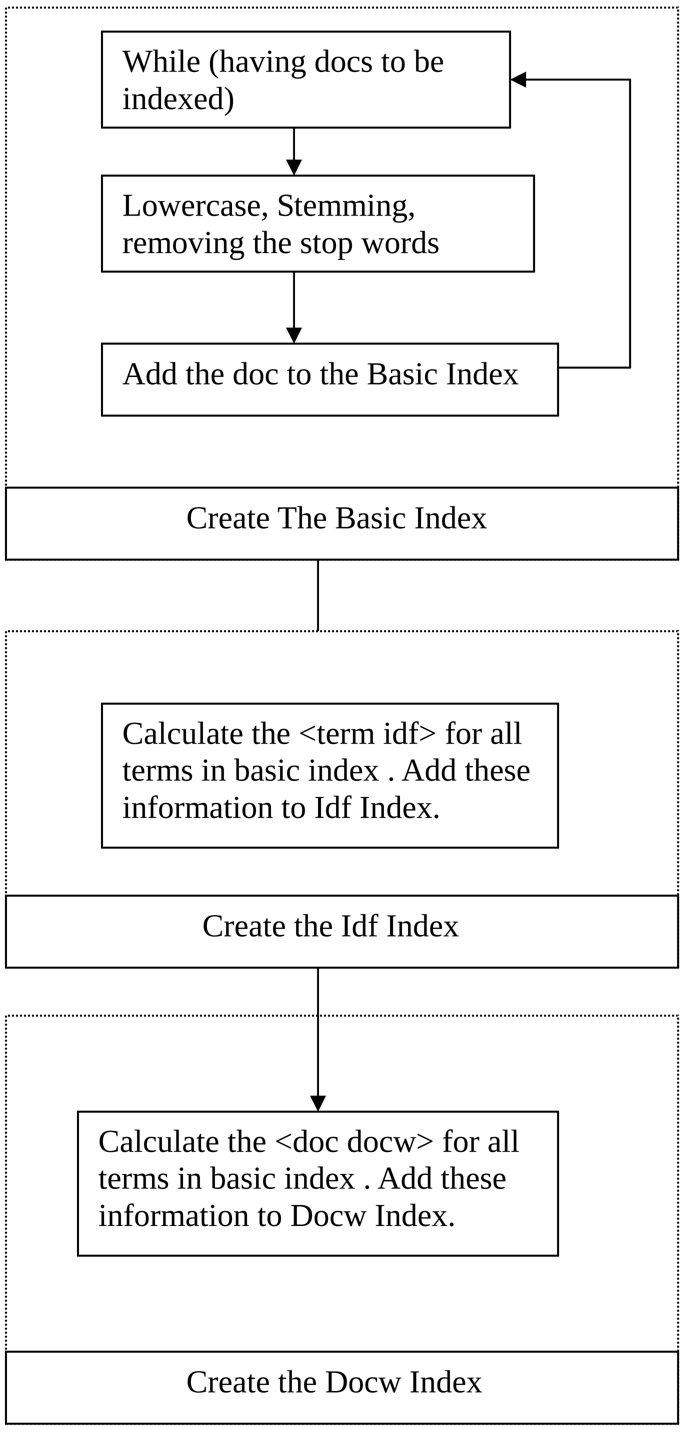

While (having docs to be indexed)
Lowercase, Stemming, removing the stop words
Add the doc to the Basic Index
Create The Basic Index
Calculate the <term idf> for all terms in basic index . Add these information to Idf Index.
Create the Idf Index
Calculate the <doc docw> for all terms in basic index . Add these information to Docw Index.
Create the Docw Index

Fig. 11. Indexing Execution Flow of Ai-Times

4.2 Improve VSM using Query Expansion and Term Reweighting

Ai-Times's retrieval model is based on VSM, for all fetched web documents, an m-dimensional vector is generated for each web document from sets of terms with associated weights, where m is the number of unique terms in the document collection. A cosine similarity measure is used to compute the similarity between a document doc_j and a query q.

$$\text{Similarity (q, doc}_j) = \frac{\sum_{k=1}^{m} (w_{kj} \bullet v_k)}{\sqrt{\sum_{k=1}^{m} (w_{kj})^2} \bullet |Q|}$$

$$w_{kj} = f_{kj} \bullet Log(D / d_j)$$

$$v_k = f_k \bullet Log(D / d_j)$$

where:

w_{kj} is the term weight for keyword term k in document j.

f_{kj} is the frequency in which the term j occurred in the document j.

d_j is the number of documents containing the term k, and,

D is the total number of documents in the set.

v_k is the term weight for keyword term k in the query.

f_k is the frequency in which the term j occurred in the query.

Q is the query weight.

However, the problem of mismatch is becoming more and more common. People often use different words to describe their concepts. A novel solution to this problem is query expansion [4, 26]. Using query expansion, an initial search is made by the system with a user-provided query and the user then indicates which of the returned documents are relevant and which are irrelevant. The system then expands the initial query using words with similar meaning to those in the query. The expanded query often returns improved result though need additional user adjustment [26].

Instead of the user manually adjustment, an alternative approach is to expand the query by automatically analyzing the initially returned result. This approach, which is called pseudo-relevance feedback, probably consisting of automatically query expansion, term reweighting or both of them, has been proved to be useful and efficient [3, 31].

In Ai-Times, we use pseudo-relevance feedback to improve the initial query formulation by applying both query expansion and term reweighting. We expand the query firstly and then reweight the terms in the expanded query. Either the query expansion or the term reweighting has been proved to be efficient [3, 31]. Ai-Times uses both of them and the performance is relatively better than either single one. We will show the comparison result in Section 4.3.

When user submits an initial query to Ai-Times, system retrieves the top 10 documents with highest relevance measuring score and the top 10 documents with lowest relevance measuring score based on simple vector space model cosine similarity measure. The top 10 documents with highest relevance measuring score are assumed to be relevant and the top 10 documents with lowest relevance measuring score are assumed to be irrelevant. After the initial retrieval, system automatically analyzes the returned documents and then applies query expansion and term reweighting on the original query.

Figure 12 shows the query operation flow in Ai-Times. System firstly process the query on stemming and removing stop words, then retrieve the top N documents with highest similarity score , apply query operation and then do the retrieving again.

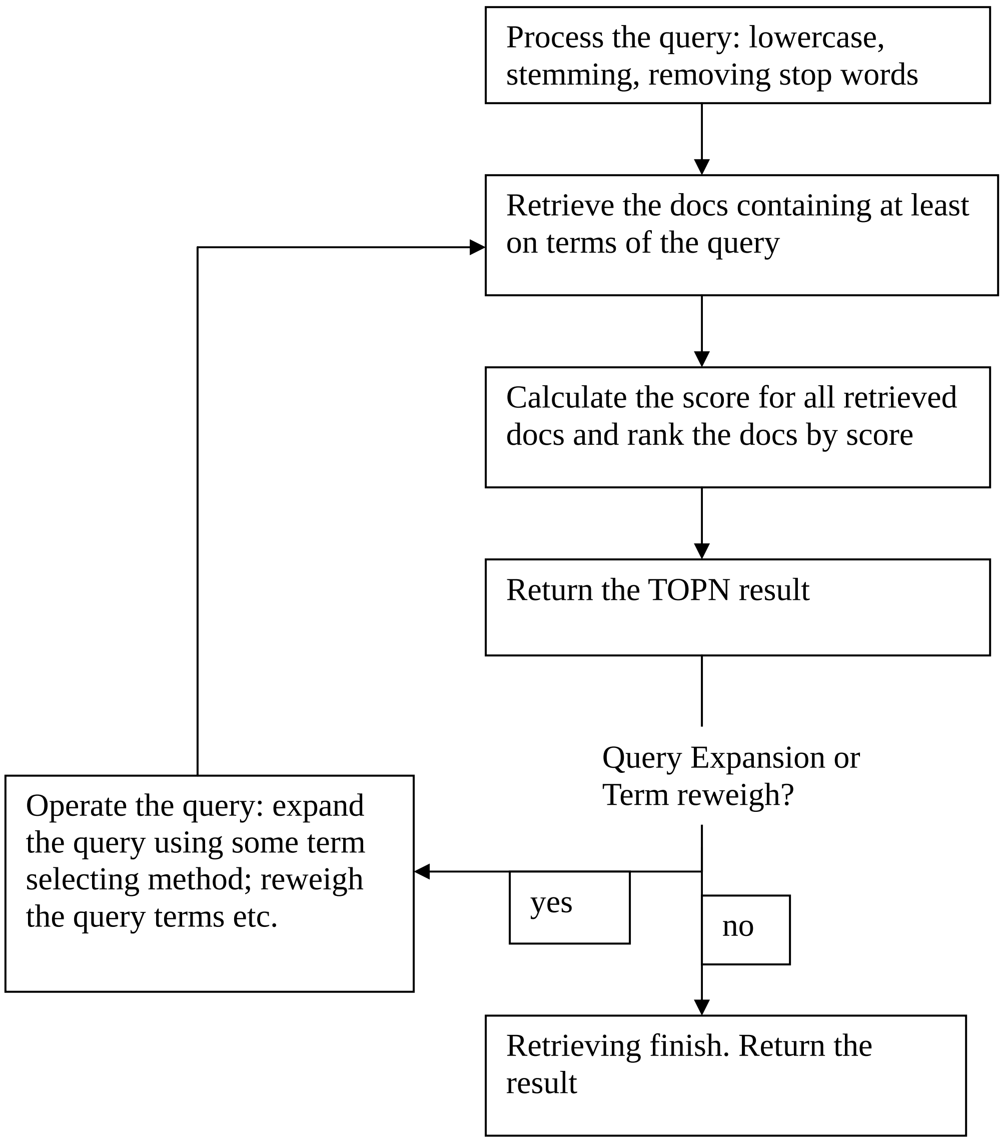

Process the query: lowercase, stemming, removing stop words
Retrieve the docs containing at least on terms of the query
Calculate the score for all retrieved docs and rank the docs by score
Return the TOPN result
Query Expansion or Term reweigh?
Operate the query: expand the query using some term selecting method; reweigh the query terms etc.
yes
no
Retrieving finish. Return the result

Fig. 12. Query operation flow in Ai-Times

4.2.1 Query Expansion

Under the bag of words model, if a relevant document does not contain the terms that are in the query, then that document will not be retrieved. The aim of query expansion is to reduce this query/document mismatch by expanding the query using words or phrases with a similar meaning or some other statistical relation to the set of relevant documents.

As previously stated, Ai-Times retrieves the top 10 documents with highest relevance measuring score based on simple vector space model cosine similarity measure. The top 10 documents with highest relevance measuring score are assumed to be relevant. Then we select the terms that are most relevant to the query and add these terms to expand the initial query.

We rank all the terms in retrieved documents using following formula:

$$tw_j = \sum_{k=1}^{n_k} w_{kj} - \sum_{i=1}^{n_i} w_{ij}$$

where:

tw_j is the weight we used to rank the terms

n_k is the number of the relevant documents containing term j.

n_i is the number of the irrelevant documents containing term j.

w_{kj} is term weight of term j in kth relevant document

w_{ij} is term weight of term j in ith irrelevant document.

After the query expansion, the initial query $< \mathrm{term}_1 \ldots \mathrm{term}_N >$ is expanded to the expanded query $< \mathrm{term}_1 \ldots \mathrm{term}_N \ldots \mathrm{term}_{N+K} >$. Here is a simple example of the query expansion:

The initial query: Reporting on possibility of and search for extra-terrestrial life/intelligence?

The expanded query: extraterrestrials, planetary society, universe, civilization, planet, radio signal, seti, sagan, search, earth, extraterrestrial intelligence, alien, astronomer, star, radio receiver, nasa, earthlings, galaxy, life, intelligence, meta receiver, radio search, discovery, northern hemisphere, national aeronautics, jet propulsion laboratory, soup, space, radio frequency, radio wave, klein, receiver, comet, steven spielberg, telescope, scientist, signal, mars, moises bermudez, extra terrestrial, harvard university, water hole, space administration, message, creature, astronomer carl sagan, intelligent life, meta ii, radioastronomy, meta project, cosmos, argentina, trillions, raul colomb, ufos, meta, evidence, ames research center, california institute, history, hydrogen atom, columbus discovery, hypothesis, third kind, institute, mop, chance, film, signs.

4.2.2 Term Reweighting

The advantages of the traditional query expansion are simplicity and good results. However, due to the simplicity, the term occurrence pattern is not considered explicitly. To supplement the query expansion techniques, we apply classic reweighting formula Rocchio [25] to modify the term weight of the expanded query.

Same as the query expansion, we reweight the terms by automatically analyzing the returned result. We submit the expanded query to the system and get the top 10 documents with highest relevance measuring score and the top 10 documents with lowest relevance measuring score. As stated before, the top 10 documents with highest relevance measuring score are assumed to be relevant and the top 10 documents with lowest relevance measuring score are assumed to be irrelevant. After the second round automatically retrieval, based on the returned documents, we increase weights of terms from relevant documents and decrease weight of terms from irrelevant documents. The query is reweighed using classic Rocchio [25] algorithm:

$$tw_j = ow_j + \sum_{k=1}^{n_k} w_{kj} - \sum_{i=1}^{n_i} w_{ij}$$

where:

tw_j is new weight of the term in the query

n_k is the number of the relevant documents containing term j.

n_i is the number of the irrelevant documents containing term j.

w_{kj} is the term weight of term j in kth relevant document

w_{ij} is the term weight of term j in ith irrelevant document.

ow_j is the initial weight of the term in the query.

After term reweighting, the expanded query $< \mathrm{term}_1 \ldots .\mathrm{term}_N >$'s weight vector $< ow_1 \ldots .\mathrm{ow}_N >$ will be reweighed to weight vector $< \mathrm{tw}_1 \ldots .\mathrm{tw}_N >$.

4.3 Experiments and results

4.3.1 Overview

In the experiment, we indexed 64814 documents; the data size is around 419 Mb.

After indexing, the size of 3 sub-indexes is:

Basic Index: 177 MB

Idf Index: 7.77 MB

Docw Index: 2.74 MB

Ai-Times retrieves the top 10 documents with highest relevance measuring score and the top 10 documents with lowest relevance measuring score based on simple vector space model cosine similarity measure. The top 10 documents with highest relevance measuring score are assumed to be relevant and the top 10 documents with lowest relevance measuring score are assumed to be irrelevant.

4.3.2 Evaluation Measures

Recall and precision are used to evaluate the performance of the retrieval model.

I. Recall

A measure of the ability of a system to present all relevant items.

recall =number of relevant items retrieved/number of relevant items in collection

II. Precision.

A measure of the ability of a system to present only relevant items.

precision =number of relevant items retrieved/total number of items retrieved.

Precision and recall are set-based measures. That is, they evaluate the quality of an unordered set of retrieved documents. To evaluate ranked lists, precision can be plotted against recall after each retrieved document as shown in the example below. To facilitate computing average performance over a set of topics, each with a different number of relevant documents, individual topic precision values are interpolated to a set of standard recall levels (0 to 1 in increments of .1).

The particular rule used to interpolate precision at standard recall level I is to use the maximum precision obtained for the topic for any actual recall level greater than or equal to i. Note that while precision is not defined at a recall of 0.0, this interpolation rule does define an interpolated value for recall level 0.0. In the example, the actual precision values are plotted with circles (and connected by a solid line) and the interpolated precision is shown with the dashed line.

4.3.3 Results

The interpolated recall-precision chart shows the improvement of query operation.

Figure 13 shows the Interpolated Recall - Precision Averages information of the traditional VSM:

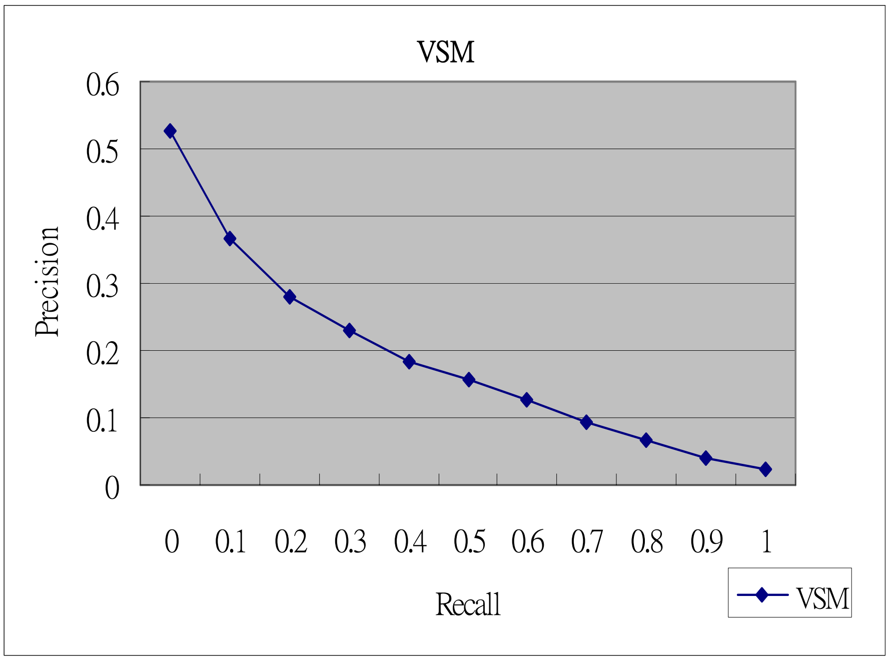


Fig. 13. Interpolated Recall - Precision Averages chart of VSM

Figure 14 shows the Interpolated Recall - Precision Averages information of the VSM+Query Expansion model:

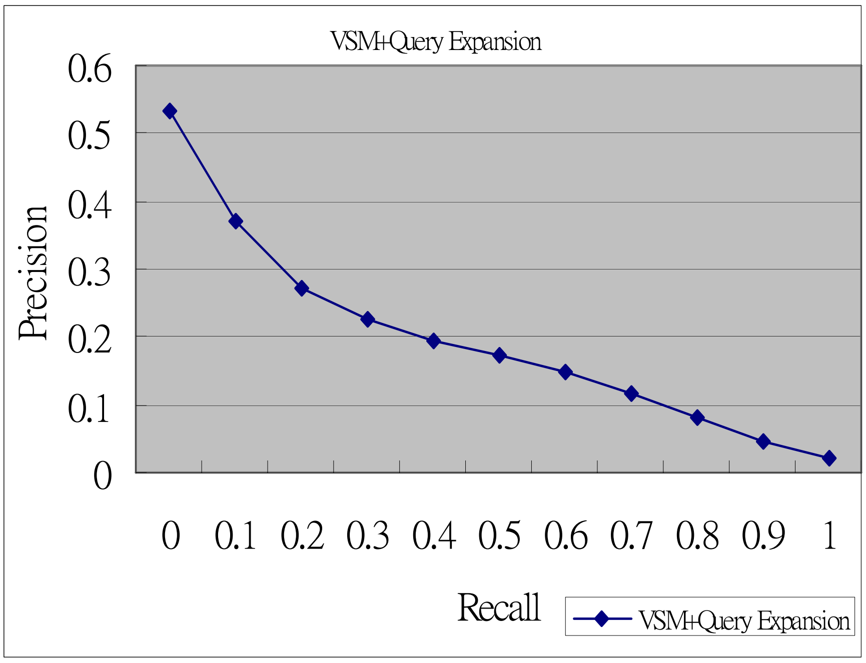


Fig. 14. Interpolated Recall - Precision Averages chart of VSM+Query Expansion

Figure 15 shows the Interpolated Recall - Precision Averages information of VSM+Term Reweighting model:

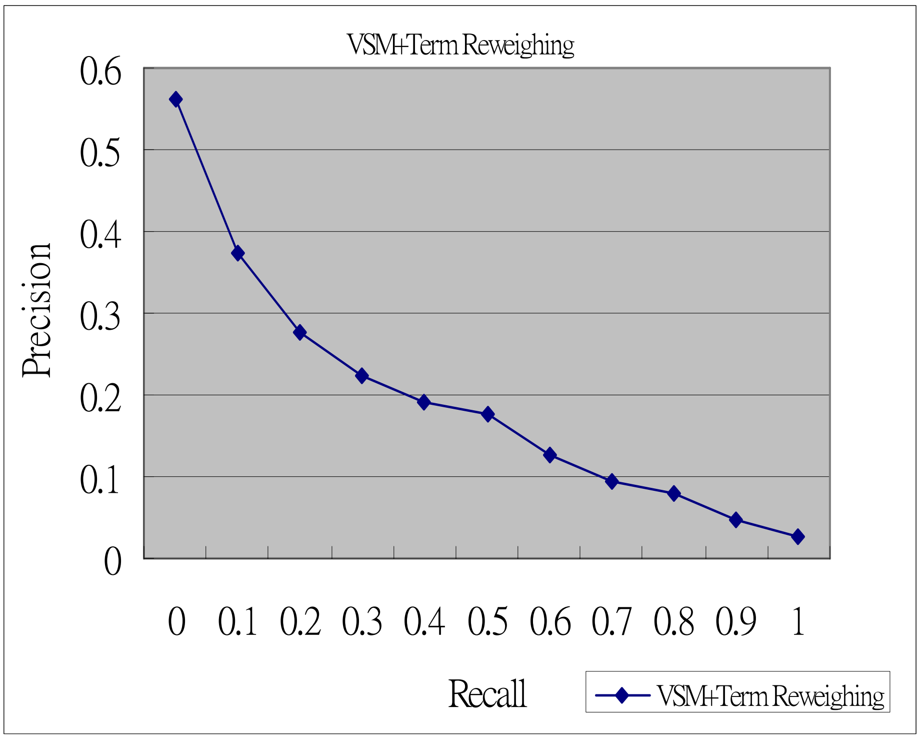


Fig. 15. Interpolated Recall - Precision Averages chart of VSM+Term Reweighting

Figure 16 shows that the performance of VSM+Query Expansion+Term Reweighting is the best in the total 4 models.

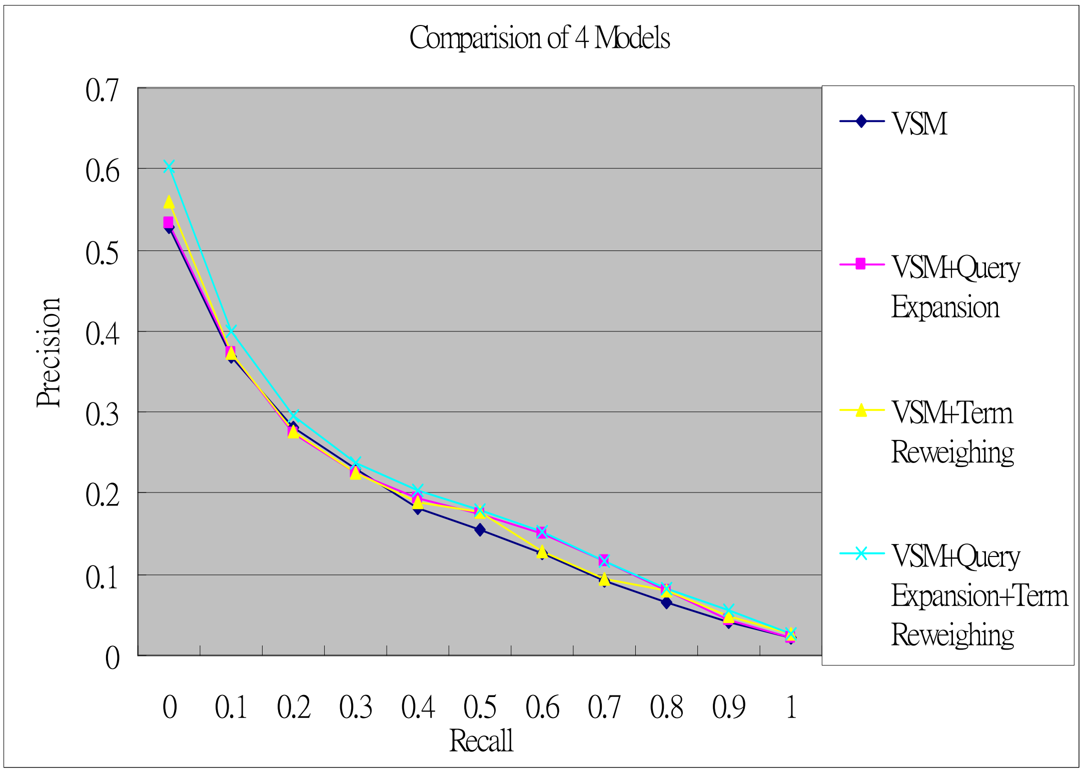


Fig. 16. VSM+Query Expansion+Term Reweighting

From these figures we can see that both VSM+Query Expansion and VSM+ Term Reweighting models have better performance than the pure VSM model. However, when compared with the VSM+Query Expansion+Term Reweighting model, for any given the Recall value, the VSM+Query Expansion+Term Reweighting always has better Precision. Therefore, it is obvious that the VSM+Query Expansion+Term Reweighting model has the best performance in all 4 models.

4.4 Conclusion

In this chapter we presented the indexing and retrieval model of Ai-Times. The Ai-Times index contains 3 sub-indexes basic index, idf index and docw index. The Ai-Times index allows us to quickly retrieve the news for users. We use query expansion and term reweighting to improve the traditional VSM model. Section 4.3 shows that the modified VSM model has best performance in total 4 models.

In next chapter we will discuss how MapReduce is used in Ai-Times to parallel all the modules and how we modify the traditional MapReduce to get better performance.

Chapter 5

Parallel Strategy

5.1 Traditional Parallel Strategy

As the size of the web grows; it is difficult for a single crawler node to finish downloading pages in a reasonable amount of time. In order to address this problem, we extended the crawler to parallel architecture [21] in stage 2 and providing details below.

Parallel crawler has been well studied [7, 12, 29]. A parallel crawler has many advantages compared to a single node crawler [12]. Junghoo Cho [12] compared multiple architecture of parallel crawlers and clarified the relative metric of each architecture. However, the general-purpose parallel crawler's architecture is not applicable to news retrieval problem. A more general drawback of parallel approaches for news retrieval however is that they still require that the crawler revisit all of the documents rather than just those documents which are of interest to users or which are most likely to have been updated since the last visit. We present our parallel crawler's architecture and then discuss its characteristics.

Figure 17 illustrates the architecture of Ai-Times parallel crawler. Ai-Times parallel crawler consists of several crawler nodes which can cooperate well in fetching the web documents. Ai-Times parallel is using static task assignment scheme [12] to assign tasks and using batch url exchanging scheme to reduce communication overhead.

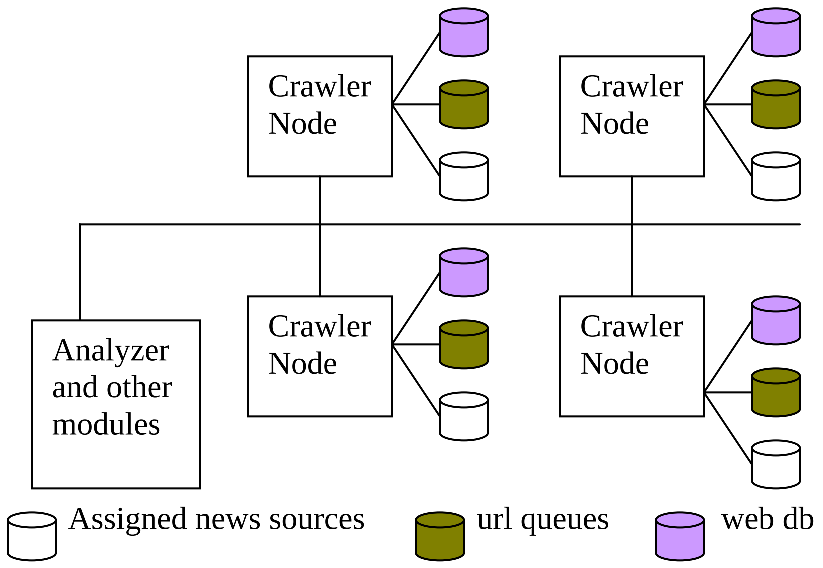


Fig. 17. Architecture of Ai-Times parallel crawler

Task assignment

When multiple crawler nodes download web pages in parallel, they need to coordinate with each other to avoid overlap downloading. In order to minimize the communication overhead, the static assignment scheme [12] is used in our parallel crawler.

In this scheme, urls in the queues are partitioned by their hash value. Each crawler node knows which url should be processed by which crawler node. When a crawler node gets a url which should be processed by other crawler node, it will put the url into a temp queue and then transfer the url to corresponding crawler node through batch url exchanging.

Url hash function

A url hash function is used to compute a web page's hash value. Based on the hash value, the web page is assigned to relative crawler node. We compute the hash value based on the site name of the web page. In this case, web pages from a same site will be allocated to a same crawler node.

Batch url exchanging scheme

When a crawler node discovers a link that points to a web page which should be processed by other crawler nodes, it will put this link into a temp url queue for a while and then transfer to relative crawler node when the number of the links in the temp queue has reached a predefined threshold. On the other hand, a crawler node is always ready to accept the request from other crawler nodes which is ready to send urls to it.

The batch url exchanging scheme reduces communication overhead and guarantees the coverage of the crawler as a whole.

5.2 Using MapReduce in Ai-Times

During the development of the traditional parallel strategy of Ai-Times, we found that the input data of many modules such as the Parsing and Extraction module, Indexing module were usually huge, thus the computation effort is better shared by multiple machines. We should extend most modules to parallel architecture. However, the coding effort, scalability and fault tolerant ability of traditional parallel algorithm are becoming bottlenecks. We should continue to improve the parallel strategy of Ai-Times.

The existing news information retrieval systems [10, 12, 29] have already discussed how to use appropriate techniques to decentralize the computation in a news retrieval system. However, little study has been done on using MapReduce in a web information retrieval system. In particular, we believe the following issues make the study of using MapReduce in Ai-Times interesting and challenging:

Scalability

The input data is usually large in an information retrieval system and the computation effort should be shared by multiple machines. With the explosion of the online web pages, the system should be easy to be extended by adding some machines into the system. This requires the system to be much scalable. At this point, a news retrieval system can benefit much from MapReduce.

Minimize coding effort

Note that the information retrieval system is usually a large-scale system. It consists of many modules such as crawler, indexing, parsing and extraction etc… In order to parallelize these tasks, distribute the fetched data and handle the failure of machines, we often implement large code which is quite complex to manage. By using MapReduce, we can hide the distribution and fault tolerance within the MapReduce library and provide an abstract interface to the programmer. Thus the code is often simpler and smaller.

Fault Tolerance

Note that failure of the machines or the hard disks is very common in an information retrieval system. Failure of a running machine or an active hard disk should not cause the whole system fail or any data lost. MapReduce is resilient to large-scale machine failures. MapReduce is based on a global file system [14], thus we can quickly recover the data when a disk fails.

5.2.1 Global File System

Note that in an information retrieval system, files are usually very huge by traditional standards. In Ai-Times [20], it is very common that a file grows to multi-GB size. The file may contain the downloaded web page objects or indexing objects.

Failure of the hard disks is very common. Once the hard disks fails, all the data stored on that component should be considered lost though we may be able to fix the failed hard disks later.

Most files are mutated by appending new data. Random recording data is prohibited.

In order to design a file system that meets Google's need, Google designs and implements a file system named Google File System [14].

Figure 18 describes the GFS architecture:

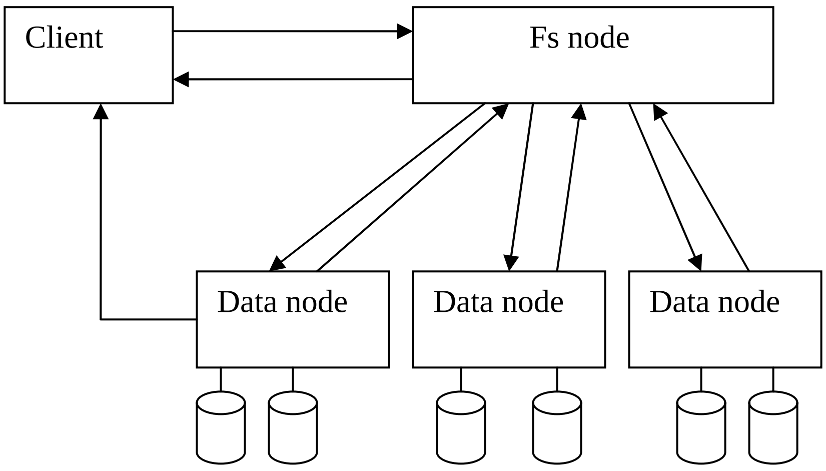


Fig. 18. GFS architecture

Global file system is a pre-condition of using MapReduce in a search engine. Therefore, Ai-Times has a very simple implementation of global file system based on which we can implement MapReduce. Ai-Times' global file system is a simple implementation of google file system and does not have the garbage collection and diagnostic tools which are all contained in google file system. It just basically satisfies our requirement to implement a MapReduce based news retrieval system.

Ai-Times' global file system consists of a single master node (also named as Fs node) which is responsible for managing the files name space which basically contains a table describing which file consists of which set of chunks. The data structure in the Fs node is simple:

filename_1 --> chunkID_11, chunkID _12, ... chunkID _1X1.

..

filename_k --> chunkID_k1, chunkID _k2, ... chunkID _kXk.

where:

finename_k is a string identifying the kth file’s name. The file contains Xk chunks which are identified by chunkID_k0, chunkID _k1, ... chunkID _kXk. As the file grows, the number of the chunks associated with the file increases.

Data nodes are responsible for storing data:

chunkID_ij --> [array of bytes]

As stated above, files in Ai-Times’ global file system are divided into fixed-size chunks. Each chunk is identified by a unique global id.

A given chunk should have at least 2 copies stored on multiple data nodes in case the failure of any data node. A given Data node has at most one copy of a given chunk, and will often have no copies.

5.2.2 MapReduce Architecture Design

The input data of the Map function will be partitioned into M splits associated with the number of the available nodes. These spits can be processed in parallel on distributed Map task nodes.

The size of each split can be determined by the requirement and related implementation. In Google, the size is between 16MB-64MB [14].

Reduce tasks are also distributed by partitioning the intermediate key space into a number of pieces. At the end of the computation, all the reduce results will be collected as whole output.

The Execution flow is straightforward shown in Figure 19：

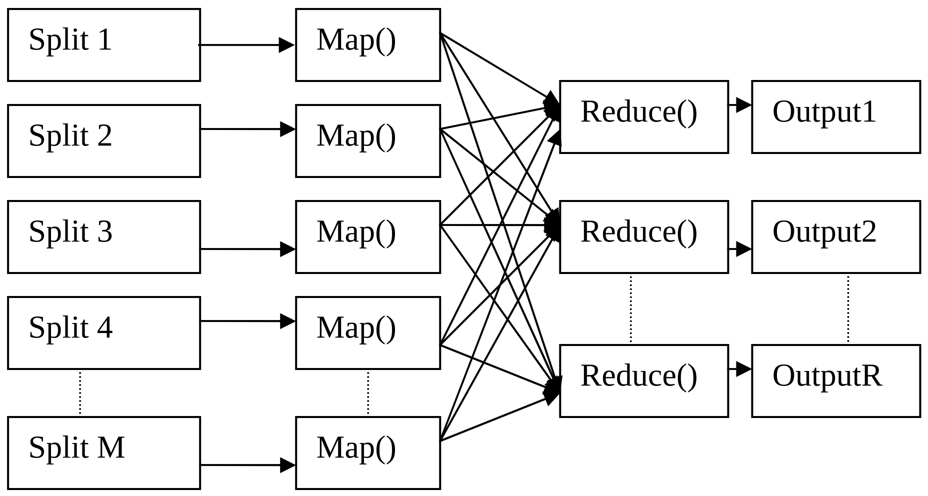


Fig. 19. Execution flow of MapReduce

Figure 19 shows the execution flow of a whole computation round. Input data is partitioned into M pieces and is then passed to M Map nodes. Each Map node processes the input pairs, generates the intermediate pairs which are partitioned into R pieces according to some hash function on the intermediate key. Both the Map node and the Reduce node are task execution node. However, in order to make the system have the knowledge about splitting the input data, assigning the tasks and managing the node state, we should have a node which is responsible for the tasks stated above. Google call this kind of nodes “Master”.

There are three types of nodes in the MapReduce system now. The map node and the reduce node are responsible for processing map task and reduce task respectively. The master node stores all map and reduce nodes’ state and the idle nodes’ state.

Note that the MapReduce programming is based on GFS [14]. As stated in chapter 5.2.1, there are other two types of nodes: FS node and data node. Therefore, the architecture of MapReduce can be shown in Figure 20:

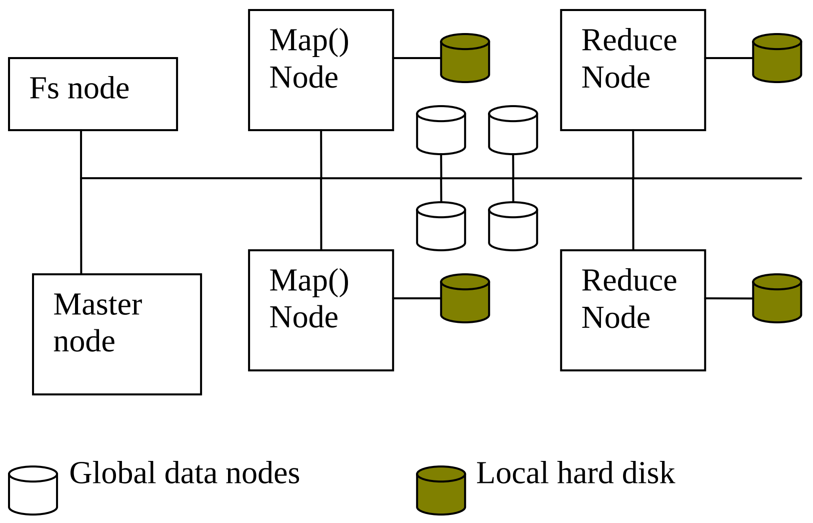


Fig. 20. Architecture of MapReduce

5.2.3 Fault Tolerance

As stated before, there are totally 5 types of nodes in the MapReduce system: Master node, Map node, Reduce node, Fs node and data node. Also we have some idle nodes which are just waiting for scheduling from the Master node and doing nothing else. If an idle node is scheduled by the Master node, it becomes some type of node stated above.

The Master node stores all map and reduce nodes' state and the idle nodes' state. Master knows how to partition the input pairs and the intermediate pairs, assigns Map and Reduce tasks to related idle nodes, and keeps all tasks' state. It asks another idle node to re-execute a task once that task execution fails.

Map node and Reduce node execute Map and Reduce tasks respectively. They communicate with the Master node periodically to let Master know their up-to-date state. They also send heart beat message to master every five seconds (The interval is configurable). If the Master has not received the heart beat message from a task node for more than 60 seconds, Master marks the task executing on that node as failed and asks another idle node to re-execute the failed task.

Data node failure

It should be clear that the system is mostly available and reliable when chunks are available on many different Data nodes. A Data node failure does not mean the data will disappear. Once a data node fails, the FS node simply passes the client's requests to other nodes containing the backup chunks and replicates the failed chunks on other data node.

FS node failure

The FS node is a critical failure node. If the FS node fails, the whole system is unavailable. However, the FS node needs to do little actual work so that the load on it is not high. In order to handle the FS node failure scenario, the system should backup FS node's data periodically.

Map node or Reduce node failure

It is common in a MapReduce system that a task node fails its task. The task node uploads to the Master node its task's state periodically. It also sends periodic heart beat message to Master node. Once the Master does not receive the heart beat message from a task node, Master marks the task as failed and put the task in the task list for rescheduling.

Master node failure

Similar with the FS node, the Master is also a critical failure node. In Google's implementation, MapReduce system will abort the computation once the master node fails. However, this is not a good idea when the MapReduce system is going to derive some important data. One solution is to build a backup master node whose data and state is synchronized with the Master node. Any insert, update, or delete operation on the master node's state tree will trig a same operation on the backup master node.

Using a backup master node loses some efficiency of the whole system because of the synchronization. However, consider the fault tolerance ability; it is worth adding a backup master node when the MapReduce system is used for some critical jobs.

5.2.4 Using MapReduce in Parallel Crawler

We present an optimized distributed crawler [21] in stage 2 of Ai-Times. The static assignment scheme [12] is used in task assignment. The batch url exchanging is used to reduce communication overhead and guarantee the coverage of the crawler as a whole. However, when used in more than 10 servers, the program begins very complex to manage. To fetch more than 500 millions pages is impossible.

Using MapReduce in crawler module makes the program more simple and scalable and makes the task to fetch more than 1000 millions possible. Figure 21 shows the Execution flow of crawler using MapReduce.

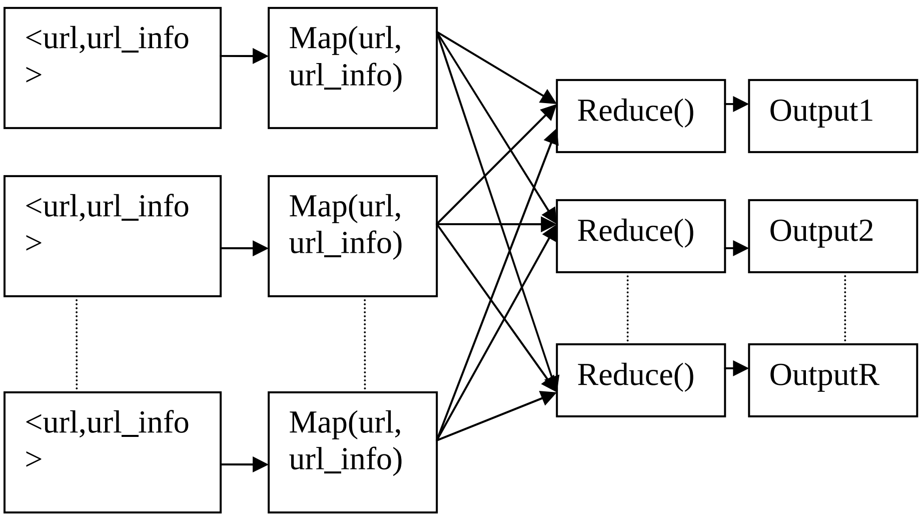


Fig. 21. The Execution flow of crawler using MapReduce

The Map and Reduce functions of crawler module:

```
Map (String url, String url_info):
    // url: the url of the web page to be fetched.
    // url_info: related knowledge of the url.
    if the web page is fetched successfully:
    EmitIntermediate(url, content);
Reduce is identity
```

Reduce is identity means that the Reduce function will not do any operation on the intermediate pairs and just return the intermediate pairs as the final result. In this case, Reduce in the Crawler module just outputs the intermediate pairs (url, content) as final result.

5.2.5 Using MapReduce in Distributed Parsing and Extraction

Content parsing and extraction is a time consuming task because the input data is usually very huge. In stage 2, Ai-Times never distributes this kind of tasks so the parsing and extraction is usually a bottleneck of distributable processing.

Using MapReduce to distribute the parsing and extraction will by no means eliminate this bottleneck. Figure 22 shows the Execution flow of parsing and extraction module using MapReduce:

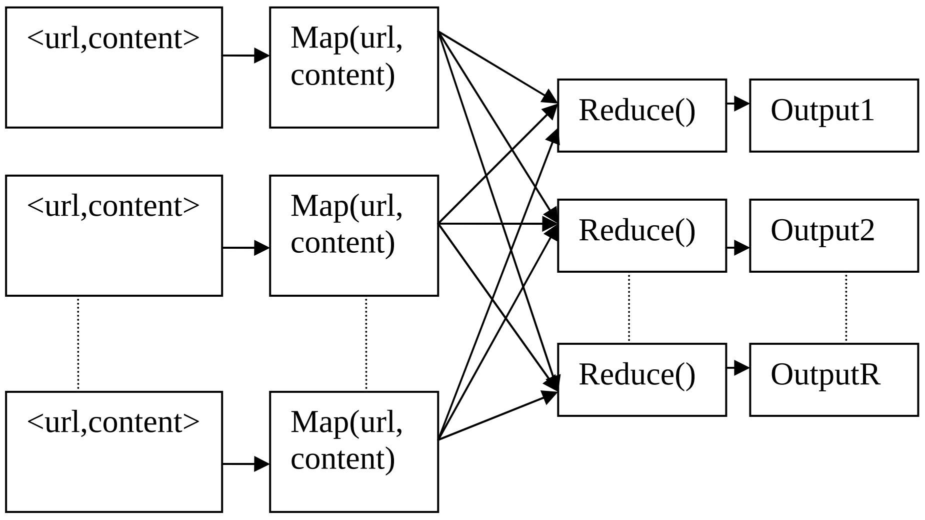


Fig. 22. The Execution flow of parsing and extraction module

The Map and Reduce functions of parsing and extraction module:

```
Map (String url, String content):
    // url: the url of the fetched web page.
    // content: the html content of the fetched web page.
        If the web page is parsed correctly:
          EmitIntermediate (url, pure_content);
    EmitIntermediate (url, outlinks);
   // pure_content: The content of the web page after parsing and
extraction.
   // outlinks: The list of the out links originated from the web page.

Reduce is identity
```

Same as the crawler module, Reduce in the Parsing and Extraction module is identity, just outputs the intermediate pairs (url, pure_content) and (url, outlinks) as final result.

5.2.6 Using MapReduce in Distributed Updating Web Db

After crawling, parsing and extraction, we should update the old web db using the output data derived from previous two steps for example, the outlinks data. Similarly, the input data of the update operation is huge so MapReduce is also quite useful here. Figure 23 shows the Execution flow of Update module using MapReduce:

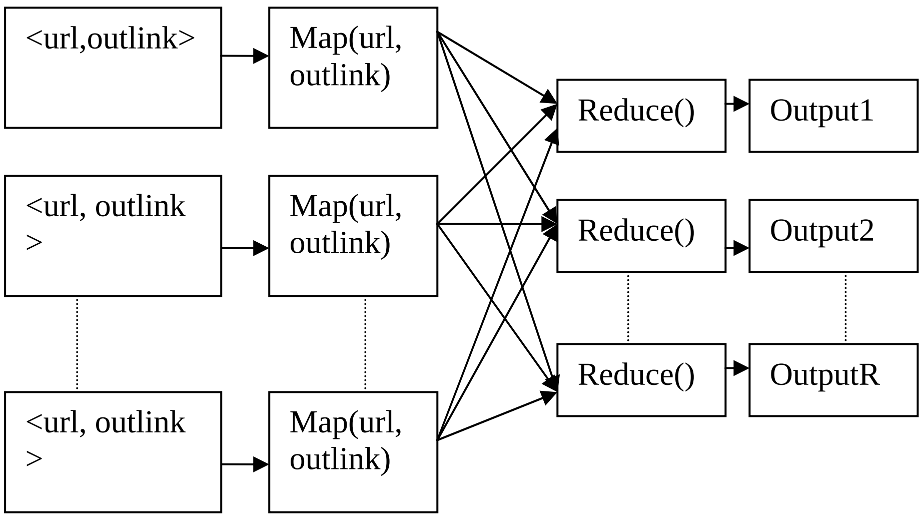


Fig. 23. the Execution flow of Update module using MapReduce

The Map and Reduce functions of update module:

```
Map (String url, List outlinks):
    // url: the url of the web page.
    // outlinks: The list of the out links originated from the web page.
    if the web page is parsed correctly:
        EmitIntermediate (url, url_info);
    // url_info: related knowledge of the url.

Reduce (String url, Iterator url_infos):
    // url: a url
    // url_infos: a list of url_info
    Url_Info url_info= null;
```

```
        for each v in url_infos:
           url_info.update(v);//update the url_info
        Emit(url_info);
```

5.2.7 Using MapReduce in Distributed Indexing

The most significant use of MapReduce in Google has been a complete rewrite of the indexing system [13]. However, Google Inc. never describes much implementation or algorithm details of Google distributed indexing module. In Ai-Times, the Execution flow of the distributed indexing module using MapReduce can be shown as below:

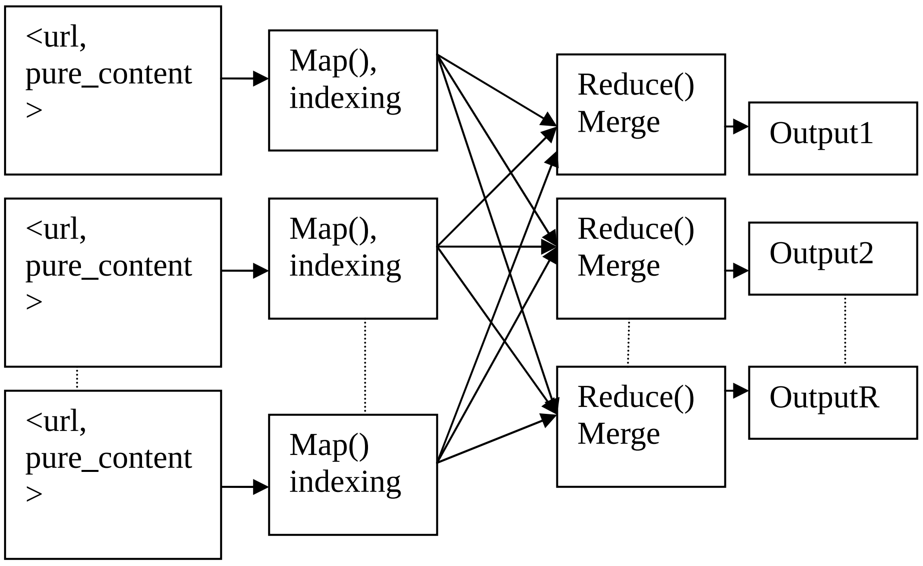


Fig. 24. the Execution flow of Indexing module using MapReduce

The Map and Reduce functions of indexing module:

```
Map(String url, String pure_content):
    // url: the url of the web page.
    // pure_content: The content of the web page after parsing and
extraction.
    for(each word v in pure_content)
        EmitIntermediate(v, position_info);
    //position_info: The position information related to the word v.

Reduce(String word, Iterator position_infos):
    // word: a word.
    // position_infos: a list of position_info.
    for each v in position_infos:
            position_info_list= position_info_list+”;”+ v.
      Emit(position_info_list);
```

As stated above, the Map function accepts the <url, pure_content> pairs as input data. Map function segments the pure_content by individual word and then output the <word, position_info> pairs as intermediate result. The Reduce function merges the intermediate pairs and generates the <word, position_info1; position_info2..position_infok> as the final result.

5.3 Task Executing Time Estimation Algorithm

In a MapReduce system, when a job is partitioned into multiple tasks and the tasks are executed in parallel, the completion time of the job is determined by the last finished task. In a distributed indexing module using MapReduce, it is common that one of the map task nodes takes unusually longer time to finish its task. The whole system is waiting that node’s result for Reduce use. A clever mechanism is to start a backup execution of the remaining tasks in-progress when the job is close to completion. The task will be marked as finished when the primary task or backup task finish. This will help in most scenarios. Figure 25 shows how backup tasks work.

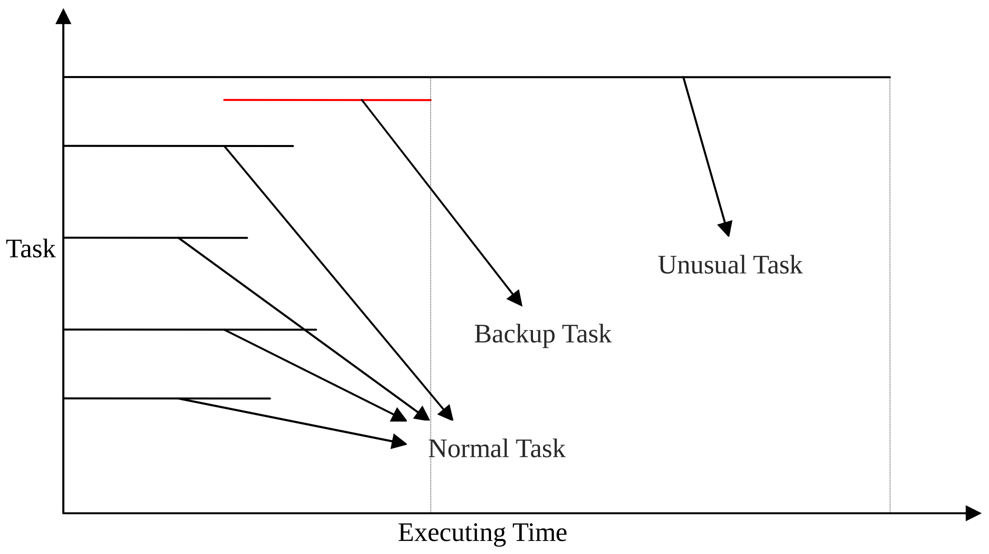


Fig. 25. Scenario of Backup Task Will Help

However, in our experiences of using MapReduce, the backup task mechanism doesn’t always work. Sometimes the task completion time is completely determined by the input data. Starting backup tasks of this kind of tasks won’t shorten the completion time at all. A recent problem we experienced is the parallel crawler using MapReduce, the input data is partitioned by the hash value of the url’s hostname. Thus, a task node is responsible for fetching web pages from a set of websites (Assume 1 website here for simplicity). If a website is hard to connect due to the bandwidth or server problems, the task node which is responsible for fetching the pages of that website (page is assigned to task nodes based on its host hash value, so pages from the same website will be assigned to the same task node) will take long time to complete the task and starting backup tasks doesn’t make much sense, which can be shown in Figure 26:

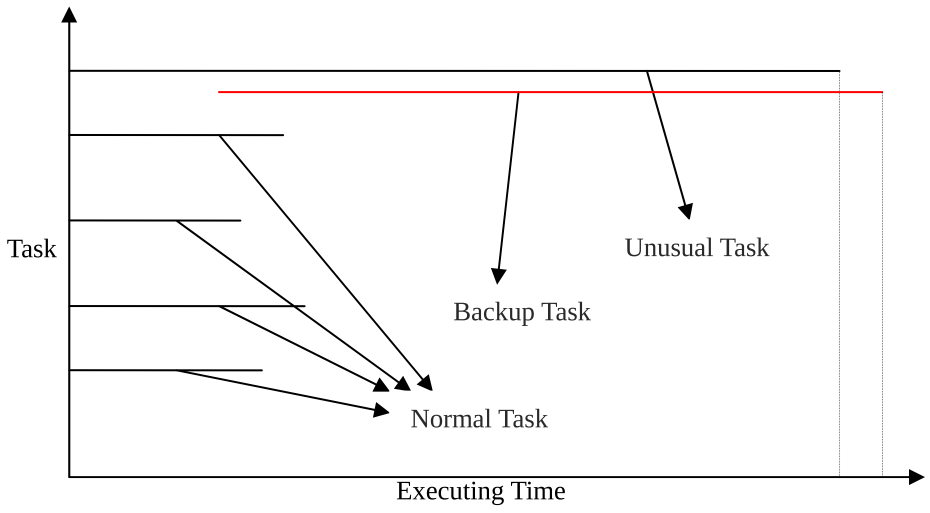


Fig. 26. Scenario of Backup Task Fails

The modified MapReduce architecture deals with the problem of excessive task time consumption by taking into account the task completion time when partitioning the input data. To estimate the task completion time, we used an algorithm based on historical data which we call the Task Estimation algorithm.

The modified algorithm is as follows:

Algorithm III: Task Estimation Algorithm

Assume we have N1 data records to be partitioned. Each record has T attributes: attribute1….attributeT. From the past MapReduce processing, we get N2 data records' processed time.

The data structure of the record looks like this:

```
Record{
attribute1
attribute2
……
attributeT
processedtime //null if the record has not been processed
estimatedtime
}
```

Records can be partitioned into C_t categories according to the t^{th} attribute's value. In the same category, the t^{th} attribute's values of the records are same or similar. In Ai-Times, the String attribute or the discrete attributes are partitioned by their hash value; the continuous attributes are partitioned by their range; C_t is configurable.

We calculate the average executing time of the records in each category:

$$\text{AverageTime (tj)} = \frac{\sum_{i=1}^{n_{tj}} record\,(i).processedtime}{n_{tj}} \tag{1}$$

where:

AverageTime (tj): average processedtime of the records which belongs to the j^{th} categories according to the t^{th} attribute.

n_{tj}: total number of the records belonging to the j^{th} categories according to the t^{th} attribute.

$$\overline{\text{AverageTime (t)}} = \frac{\sum_{j=1}^{C_t} AverageTime(tj)}{C_t} \tag{2}$$

where:

C_t: the number of the categories according to the t^{th} attribute.

Then we calculate the variable Determine of each attribute using following formula.

$$\text{Determine (t)} = \sqrt{\frac{\sum_{j=1}^{C_t}\left(AverageTime(tj) - \overline{AverageTime(t)}\right)^2}{C_t}} \tag{3}$$

The attribute having the larger Determine value is considered to goven the task executing time more directly. Among the T attributes, we selected the top K attributes which are considered to eventually determine the task executing time. In Ai-Times, we make the k to be 1.

Then we update the records' estimatedtime using the top K arrtibutes' average time value, which we get from formula (1)

$$\text{record.est imatedtime} = \frac{\sum_{i=1}^{K} AverageTim\ e(ij)}{K} \tag{4}$$

where:

K is the number of the top attributes we selected.

AverageTime(ij) is the average processedtime obtained from formula (1). The record belongs to j^{th} category according to the i^{th} attribute.

After selecting the most important K attributes. We can then partition the data records to be processed using following 3 formulas:

Compute the record's hash value to partition the data records into K splits.

Hash (record[i]) mod K (5)

where:

record [i]: the ith record to be processed.

K: the number of the data splits we want to partition the data into.

Sum up all the records' estimated processing time in the same split to get the split's estimated executing time.

$$\text{TotalTime (k)} = \sum_{i=1}^{n_k} record\ (l).estimatedt\ ime \tag{6}$$

where:

TotalTime (k): the estimated executing time of the k^{th} data split.

l: the sequence of the record in the k^{th} data split.

n_k : the total number of the records in the k^{th} data split.

estimatedtime: the estimated processing time of the l^{th} record.

The deviation between any two splits' estimated executing time should be less than a predefined threshold.

$$|\text{TotalTime (i)} - \text{TotalTime (j)}| < \text{CONSTANTTIME} \qquad (7)$$

where:

TotalTime (i), TotalTime (j): the estimated executing time of the ith and jth data split respectively.

CONSTANTTIME: the predefined deviation threshold of the estimated time between any two data splits.

5.4 Experiments

5.4.1 Environment

Ai-Times is running on several PC servers. The system set up is as following:

1. Each server is configured with P4 2.4 GHz CPU and 512 MB RAM

2. The servers are connected with each other through 100 MB intranet cable.

3. Each server is connected to the internet with 1 MB independent bandwidth.

5.4.2 Fault Tolerance Ability

As stated before, there are totally 5 types of nodes in the MapReduce system: Master node, Map node, Reduce node. The Master node stores all map and reduce nodes' state and the idle nodes' state. Master knows how to partition the input pairs and the intermediate pairs, assigns Map and Reduce tasks to related idle nodes, and keeps all tasks' state. It asks another idle node to re-execute a task once that task execution fails.

Map node and Reduce node execute Map and Reduce tasks respectively. They communicate with the Master node periodically to let Master know their up-to-date state. They also send heart beat message to master every five seconds (The interval is configurable). If the Master has not received the heart beat message from a task node for more than 60 seconds, Master marks the task executing on that node as failed and asks another idle node to re-execute the failed task

The Master node is a critical failure node. Once the Master node fails, the whole system fails. We have a backup master node whose data and state are synchronized with the Master node. Any add, update, or delete operation on the master node's state tree will trigger the same operation on the backup master node. Once the Master node fails, we will use the backup master node as the main Master node.

5.4.3 Ai-Times Parallel Crawler Based on MapReduce vs. Other Parallel Crawler

As mentioned previously, many researches have been done on the parallel crawler. In particular, we mainly consider the most important properties addressed by [12].

Coverage

Coverage=c/n, where n is the number of the web pages the crawler need to visit, and c the number of the web pages actually visited. If no faults occur, Ai-Times parallel optimized crawler can achieve Coverage 100%.

Overlap

When multiple crawler nodes download web pages at the same time, it is possible that different crawler nodes download the same web page multiple times. By applying url hash function based on url's site name, It is obviously that the Ai-Times parallel optimized crawler can achieve Overlap 0.

Quality

Quality indicates a crawler's ability to download the "important" web pages. In a news retrieval system, a "news content document" or "index or list web document" is more important than a "valueless web document". We evaluate Ai-Times parallel optimized crawler's quality by approximating the recall and precision which we will discuss in the following section.

5.4.4 Use the task executing time estimation algorithm

In the scenario where the backup task algorithm does not even help, we use the task executing time estimation algorithm to estimate the task executing time, thus we can partition the input records more reasonably.

Table 1 shows the performance comparing result of traditional MapReduce vs Using Task Executing Time Estimation Algorithm MapReduce in crawler module:

	Traditional MapReduce	Modified MapReduce
Attributes selected as the Determine attribute		Hostname
Number of Input records	2.1 millions	2.1 millions
Size of the input records	3.38 GB	3.38 GB
Number of the tasks	250	250
Number of unusual tasks	21	3
Total completion time	12hrs 10mins	8hrs 01mins

Table 1: Traditional MapReduce vs MapReduce Using TETEA in crawler module.

In traditional MapReduce system's crawler module, we partition the pages based on their host value for task nodes to fetch. Thus the pages from the same website will be assigned to the same task node. Each node will be assigned same amount of pages.

If a website is hard to connect due to the bandwidth or web server problems, the task node which is responsible for fetching the web pages of that website will take long time to complete the task. Actually the bandwidth or web server problems are very common, therefore the unusual tasks in traditional system are much more than the system applied task executing time estimation algorithm.

Table 1 shows that the number of unusual tasks in traditional MapReduce is 21, but the number of the unusual tasks in our modified MapReduce is only 3. As a result, the total completion time of the modified MapReduce is reduced from 12 hours 10 minutes to 8 hours 1 minute. It is obvious that the modified MapReduce improve the performance.

	Traditional MapReduce	Modified MapReduce
Attributes selected as the Determine attribute		Size of the page/document to be indexed
Number of Input records	1.5 millions	1.5 millions
Size of the input records	41 GB	41 GB
Number of the tasks	200	200
Number of unusual tasks	6	1
Total completion time	3hrs 3mins	2 hrs 45mins

Table 2: Traditional MapReduce vs MapReduce Using TETEA in indexing module.

In traditional MapReduce indexing module, we assign the pages/documents for task nodes to index based on their host hash value which is the same as crawler module. From the experiment we observed that the unusual tasks happened when a particular website has much more big files in PDF/DOC/PPT formats which are hard to parsing and indexing than other websites. The task nodes which are responsible for indexing this website then took unusual long time to complete. By using task executing time estimation algorithm, we reduced the number of unusual tasks nearly to zero.

Table 2 shows that the number of unusual tasks in traditional MapReduce is 6, but the number of the unusual tasks in our modified MapReduce is only 1. As a result, the total completion time of the modified MapReduce is reduced from 3 hours 3 minutes to 2 hours 45 minutes. It is obvious that the modified MapReduce is able to reduce the number of unusual tasks and shorten the job’s total completion time.

5.5 Conclusion

In this chapter, we presented how MapReduce is used in Ai-Times and the details of using MapReduce. Also we presented how MapReduce is improved by estimating the sub-tasks’ execution time.

Experiments data show that by using MapReduce Ai-Times has good fault tolerant ability, which is important to build a robust news retrieval system. When compared with traditional parallel crawler, Ai-Times’ parallel crawler can achieve coverage 100%, overlap 0% and has relatively acceptable quality. The most significant improvement is the modified MapReduce. By using TETEA, the modified MapReduce can partition the input data more reasonably and shorten the job’s execution time by 33% in crawler module and 10% in index module respectively. However, there are still some unusual tasks in modified MapReduce system, though the number of unusual tasks is less than the traditional MapReduce. We will discuss the further research in Section 6.2.

Chapter 6

Conclusion and Future Research

6.1 Conclusion

In this thesis I have presented the research study of last 24 months. I proposed the design and the algorithms of basic modules in Ai-Times. In particular, I presented an optimized crawler algorithm, which is proven to have better performance than traditional crawler. I also provided a “Keen tag” based news extraction algorithm for extracting the news content from the news web document with nearly zero manual effort.

I have presented an improved VSM model, which is improved by query expansion and term reweighting. By using query expansion and term reweighting, Ai-Times can get better retrieval accuracy.

The most valuable contribution is that I introduced how MapReduce is used in Ai-Times to improve the scalability and fault tolerant ability and how we improve the MapReduce to get better performance. The experiments show that the improved MapReduce can help us build a scalable, efficient, and fault-tolerant distributed news retrieval system. In particular, by using TETEA, the modified MapReduce can reduce the number of the unusual tasks, which can shorten the job’s completion time obviously.

In summary, I believe that this thesis offers some useful guideline for the designers of news information retrieval system, helping them for example, design and implement a parallel online news retrieval system, optimize their crawler to be faster, implement a more flexible news content extractor which can work with less manual effort, improve the retrieval model to achieve better retrieval accuracy and improve the traditional MapReduce model to reduce the number of unusual tasks in a parallel web news retrieval system.

6.2 Future research

We believe many improvements can be done on the retrieval model or the distributed architecture of Ai-Times, for example:

Improve the retrieval model: there are some potential that we can improve the retrieval model in Ai-Times, ie, take the location in original html source into consideration when weighting the terms.

Improvement of global file system

Currently the global file system of Ai-Times is quite a simple implementation. We believe many techniques can be considered to improve the performance of the global file system, such as the garbage collection and data compression.

Locality

Higher data transfer rate is one of the disadvantages of using MapReduce. In order to conserve the network bandwidth, we can make a map task node to be a data node. Thus we can take the location of the input files into account in the Map partition stage. We can schedule a task to a task node having the task's input data on it. This can cool down the high network traffic rate.

Backup Tasks

When a job is partitioned into multiple tasks and the tasks are executed in parallel, the completion time of the job is determined by the last finished task. In a distributed indexing module using MapReduce, it is common that one of the map task nodes takes unusually longer time to finish its task. The whole system is waiting that node's result for Reduce use. A clever mechanism used in Google is to start a backup execution of the remaining in-progress tasks when the job is closed to completion. The task will be marked as finished when the primary task or backup task finish. We believe this mechanism will make sense in our system.

References

[1] S. Aggarwal and F. Hung: WIRE - A WWW-based Information Retrieval and Extraction System. Proceedings of the 9th International Workshop on Database and Expert Systems Applications, (1998), 887-892.

[2] Y. Ariki and Y. Sugiyama: A TV News Retrieval System with Interactive Query Function. Proceedings of the 2nd International Conference on Cooperative Information Systems (IFCIS), (1997), 184-192.

[3] R. Attar and A. S. Fraenkel: Local Feedback in Full-Text Retrieval Systems, Journal of the ACM, 24(3), (1977), 397 - 417.

[4] R. Baeza-Yates and B. Ribeiro-Neto: Modern Information Retrieval, Addison Wesley Longman, (1999).

[5] A. Bergholz and B. Childlovskii: Crawling for domain-specific hidden Web resources. Proceedings of the 4th International Conference on Web Information Systems Engineering, (2003), 125–133.

[6] G. E. Blelloch: Scans as primitive parallel operations. IEEE Transactions on Computers, IEEE Computer Society Press, 38(11), (1989), 1526-1538.

[7] P. Boldi, B. Codenotti, M. Santini, and S. Vigna. Ubicrawler: A scalable fully distributed web crawler. Proceedings of The 8th Australian World Wide Web Conference (AusWeb02), (2002), 711-726.

[8] C. Buckley: trec_eval IR evaluation package. Available from ftp://ftp.cs.cornell.edu/pub/smart.

[9] C. Buckley and Ellen M. Voorhees: Retrieval evaluation with incomplete information. Proceedings of 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, (2004), 25-32.

[10] M. Buzzi: Cooperative crawling. Proceedings of the 1st Latin American Web Congress. (2003), 209-211.

[11] J. Cho and H. Garcia-Molina: Effective Page Refresh Policies for Web Crawlers, ACM Transactions on Database System, 28(4), (2003), 390-426.

[12] J. Cho and H. Garcia-Molina: Parallel crawler. Proceedings of the 11th International World--Wide Web Conference, (2002), 1-13.

[13] J. Dean and S. Ghemawat: MapReduce: Simplified Data Processing on Large Clusters. Proceedings of the 6th Symposium on Operating System Design and Implementation, (2004), 137-150.

[14] S. Ghemawat, H. Gobioff and S. T. Leung: The Google File System. Proceedings of the 9th ACM symposium on Operating systems principles, (2003), 29-43.

[15] D. A. Grossman and F. Ophir: Information Retrieval: Algorithms and Heuristics; The International Series in Engineering and Computer Science, 461.

[16] K. Hu and W. S. Wong: A probabilistic model for intelligent Web crawlers. Proceedings of the 27th Annual International Conference on Computer Software and Applications, (2003), 278-282.

[17] A. J. Kurtz and J. Mostafa: Topic detection and interest tracking in a dynamic online news source. Proceedings of the 3rd ACM/IEEE–CS Joint Conference on Digital Libraries, (2003), 122-124.

[18] R. E. Ladner and M.J. Fischer: Parallel Prefix computation. Journal of ACM, 27(4), (1980), 831-838.

[19] S. J. Lim and Y. K. Ng: Categorizing and Extracting Information from Multilingual HTML Documents Database Engineering and Application

Symposium. Proceedings of the 9th International Database Engineering & Application Symposium, (2005), 415-422.

[20] N. K. Liu, W. D. Luo and M. C. Chan: Design and Implement a Web News Retrieval System. Proceedings of the 9th International Conference on Knowledge-Based & Intelligent Information & Engineering Systems (KES), (3), (2005), 149-156.

[21] N. K. Liu, W. D. Luo and M. C. Chan: A Distributed Web News Retrieval System. To appear.

[22] S. Morrison and J. A. Jose: Comparative study of online news retrieval and presentation strategies. Proceedings of IEEE 6th International Symposium, (2004), 403-409.

[23] S. Mukherjee, G. Z. Yang, W. F. Tan and I. V. Ramakrishnan: Automatic Discovery of Semantic Structures in HTML Documents. Proceedings of the 7th International Conference on Document Analysis and Recognition, (2003), 245-249.

[24] S. Renals, D. Abberley, D. Kirby and T. Robinson: The THISL system for indexing and retrieval of broadcast news. Proceedings of IEEE Signal Processing Society 1999 Workshop on Multimedia Signal Processing, (1999), 77-82.

[25] J. J. Rocchio: Relevance Feedback in information Retrieval G.11,Ed. SMART Retrieval System, Prentice Hall, (1971), 313-323.

[26] G. Salton and C. Buckley: Improving retrieval performance by relevance feedback. Journal of the American Society for Information Science, 41(4), (1990), 288-297.

[27] G. Salton, A. Wong and C. S. Yang: A Vector Space Model for Automatic Indexing, Communications of the ACM, 18(11), (1975), 613–620.

[28] M. Sanderson & C. J. van Rijsbergen: NRT - News Retrieval Tool. Electronic Publishing, EP-odd 4, (1991), 205-217.

[29] V. Shkapenyuk and T. Suel: Design and implementation of a high-performance distributed Web crawler. Proceedings of the 18th International Conference on Data Engineering, (2002), 357-368.

[30] A. Silva and E. Veloso: CoBWeb – A Crawler for the Brazilian Web, Proceedings of the String Processing and Information Retrieval Symposium & International Workshop on Groupware table of contents (SPIRE'99), (1999), 184-191.

[31] J. X. Xu and W. B. Croft: Query Expansion Using Local and Global Document Analysis, Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, (1996), 4-11.

[32] http://outgoing.typepad.com/outgoing2005/04/mapreduce.html

[33] http://www.commerce.net/wiki/MapReduce_for_Decentralized_Computation

[34] http://www.nutch.org

[35] http://www.caerwyn.com/ipn/2005/07/lab-37-geryons-mapreduce.html

[36] http://blog.commerce.net/archives/2004/11/mapreduce_for_d_1.html

[37] Search engine watch. http://www.searchenginewatch.com.

Appendix A: Experiment Result of Retrieval Model

Our retrieval model is evaluated with the trec_eval program [8, 9], below are the standard result generated by trec_eval.

VSM

```
Queryid (Num):      99
Total number of documents over all queries
      Retrieved:      99000
      Relevant:        3720
      Rel_ret:         3088
Interpolated Recall - Precision Averages:
      at 0.00         0.5269
      at 0.10         0.3676
      at 0.20         0.2814
      at 0.30         0.2289
      at 0.40         0.1818
      at 0.50         0.1558
      at 0.60         0.1269
      at 0.70         0.0928
      at 0.80         0.0654
      at 0.90         0.0416
      at 1.00         0.0222
Average precision (non-interpolated) over all relevant docs
                        0.1828


Precision:
   At       5 docs:    0.3436

   At      10 docs:    0.2861

   At      15 docs:    0.2453

   At      20 docs:    0.2244

   At      30 docs:    0.2020

   At    100 docs:    0.1095

   At    200 docs:    0.0756

   At    500 docs:    0.0419

   At 1000 docs:    0.0306

R-Precision (precision after R (= num_rel for a query) docs retrieved):

     Exact:              0.2125


Recall:

     Exact:              0.8148

     at      5 docs:     0.0871

     at    10 docs:     0.1341

     at    15 docs:     0.1643

     at    20 docs:     0.1964

     at    30 docs:     0.2283

     at 100 docs:     0.3915

     at 200 docs:     0.5252

     at 500 docs:     0.7338

     at 1000 docs:   0.8148

Average interpolated precision for all 11 recall points

     11-pt Avg:          0.1901
```

```
Average interpolated precision for 3 intermediate points (0.20, 0.50, 0.80)

    3-pt Avg:       0.1675


Fallout - Recall Averages (recall after X non-relevant docs retrieved):

    At    0 docs:    0.0608

    At   14 docs:    0.2057

    At   28 docs:    0.2363

    At   42 docs:    0.2782

    At   56 docs:    0.3144

    At   71 docs:    0.3482

    At   85 docs:    0.3736

    At   99 docs:    0.3971

    At 113 docs:    0.4186

    At 127 docs:    0.4394

    At 142 docs:    0.4596

Average recall for first 142 non-relevant docs retrieved:

                       0.3159
```

VSM+Query Expansion

```
Queryid (Num):          99

Total number of documents over all queries

      Retrieved:      99000

      Relevant:        3720

      Rel_ret:         3152

Interpolated Recall - Precision Averages:

      at 0.00          0.5340

      at 0.10          0.3722

      at 0.20          0.2726

      at 0.30          0.2256

      at 0.40          0.1948

      at 0.50          0.1737

      at 0.60          0.1499

      at 0.70          0.1154

      at 0.80          0.0808

      at 0.90          0.0443

      at 1.00          0.0221

Average precision (non-interpolated) over all relevant docs

                          0.1901

Precision:

   At     5 docs:    0.3475

   At    10 docs:    0.3212

   At    15 docs:    0.2862

   At    20 docs:    0.2545
```

```
    At      30 docs:     0.2253

    At    100 docs:     0.1300

    At    200 docs:     0.0893

    At    500 docs:     0.0518

    At 1000 docs:     0.0312

R-Precision (precision after R (= num_rel for a query) docs retrieved):

        Exact:                 0.2167


Recall:

      Exact:                 0.8482

      at      5 docs:     0.0730

      at    10 docs:     0.1291

      at    15 docs:     0.1667

      at    20 docs:     0.1893

      at    30 docs:     0.2409

      at 100 docs:     0.4139

      at 200 docs:     0.5476

      at 500 docs:     0.7398

      at 1000 docs:     0.8482

Average interpolated precision for all 11 recall points

      11-pt Avg:          0.1986

Average interpolated precision for 3 intermediate points (0.20, 0.50, 0.80)

        3-pt Avg:          0.1757
```

```
Fallout - Recall Averages (recall after X non-relevant docs retrieved):
    At   0 docs:   0.0509
    At  14 docs:   0.2006
    At  28 docs:   0.2613
    At  42 docs:   0.3068
    At  56 docs:   0.3452
    At  71 docs:   0.3710
    At  85 docs:   0.3961
    At  99 docs:   0.4207
    At 113 docs:   0.4484
    At 127 docs:   0.4666
    At 142 docs:   0.4809
Average recall for first 142 non- relevant docs retrieved:
                   0.3440
```

VSM+Term Reweighting

Queryid (Num):	99

Total number of documents over all queries

Retrieved:	99000
Relevant:	3720
Rel_ret:	3178

Interpolated Recall - Precision Averages:

at 0.00	0.5605
at 0.10	0.3732
at 0.20	0.2771
at 0.30	0.2241
at 0.40	0.1900
at 0.50	0.1770
at 0.60	0.1279
at 0.70	0.0937
at 0.80	0.0803
at 0.90	0.0479
at 1.00	0.0259

Average precision (non-interpolated) over all relevant docs

0.1921

Precision:

At	5 docs:	0.3064
At	10 docs:	0.2853
At	15 docs:	0.2620
At	20 docs:	0.2279

```
    At      30 docs:     0.2070
    At    100 docs:     0.1464
    At    200 docs:     0.0995
    At    500 docs:     0.0619
    At 1000 docs:     0.0321
R-Precision (precision after R (= num_rel for a query) docs retrieved):
        Exact:              0.2025


Recall:
      Exact:               0.8549
      at      5 docs:     0.0529
      at    10 docs:     0.0931
      at    15 docs:     0.1173
      at    20 docs:     0.1415
      at    30 docs:     0.1799
      at 100 docs:     0.3377
      at 200 docs:     0.4902
      at 500 docs:     0.7314
      at 1000 docs:     0.8549
Average interpolated precision for all 11 recall points
      11-pt Avg:          0.1979
Average interpolated precision for 3 intermediate points (0.20, 0.50, 0.80
        3-pt Avg:          0.1781


Fallout - Recall Averages (recall after X non- relevant docs retrieved):
        At     0 docs:    0.0549
        At    14 docs:    0.1816
        At    28 docs:    0.2234
        At    42 docs:    0.2725
        At    56 docs:    0.2975
        At    71 docs:    0.3364
        At    85 docs:    0.3583
        At    99 docs:    0.3942
        At 113 docs:     0.4533
        At 127 docs:     0.4777
        At 142 docs:     0.4865
Average recall for first 142 non-relevant docs retrieved:
                              0.3214
```

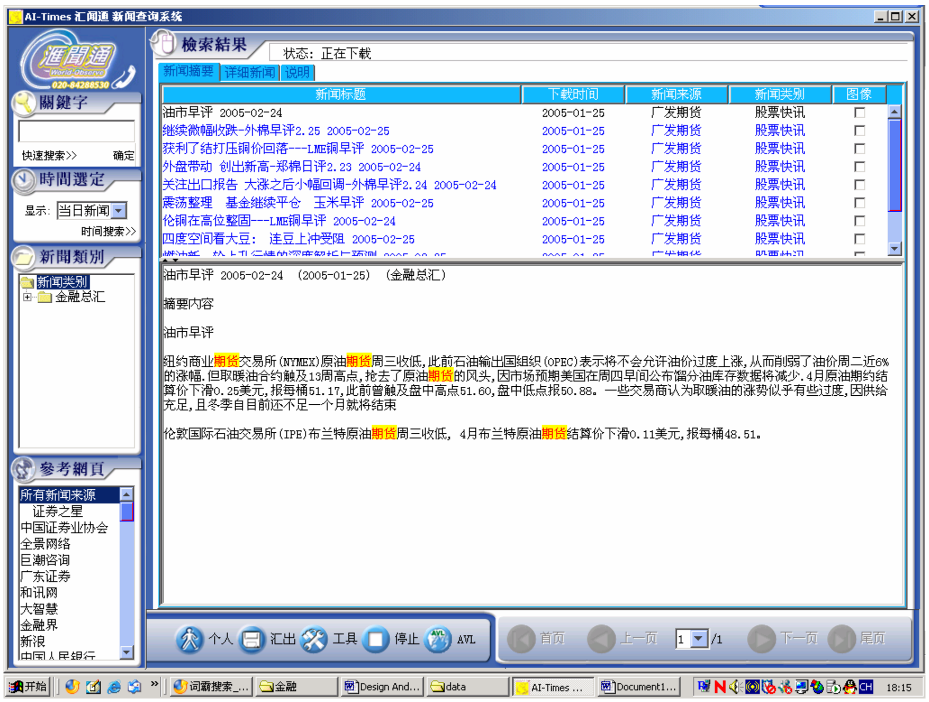
VSM+Query Expansion+Term Reweighting

```
Queryid (Num):          99
Total number of documents over all queries
    Retrieved:      99000
    Relevant:        3720
    Rel_ret:         3263
Interpolated Recall - Precision Averages:
    at 0.00         0.6027
    at 0.10         0.3995
    at 0.20         0.2947
    at 0.30         0.2366
    at 0.40         0.2029
    at 0.50         0.1798
    at 0.60         0.1537
    at 0.70         0.1153
    at 0.80         0.0816
    at 0.90         0.0569
    at 1.00         0.0272
Average precision (non-interpolated) over all rel docs
                        0.2084
Precision:
  At     5 docs:    0.3168
  At    10 docs:    0.2805
  At    15 docs:    0.2623
  At    20 docs:    0.2387
```

```
   At      30 docs:     0.1982
   At    100 docs:     0.1556
   At    200 docs:     0.0851
   At    500 docs:     0.0721
   At 1000 docs:     0.0519
R-Precision (precision after R (= num_rel for a query) docs retrieved):
        Exact:                0.2091


Recall:
      Exact:                  0.8697
      at      5 docs:      0.0535
      at    10 docs:      0.0947
      at    15 docs:      0.1294
      at    20 docs:      0.1601
      at    30 docs:      0.2000
      at 100 docs:      0.3598
      at 200 docs:      0.5100
      at 500 docs:      0.7234
      at 1000 docs:      0.8697
Average interpolated precision for all 11 recall points
      11-pt Avg:          0.2137
Average interpolated precision for 3 intermediate points (0.20, 0.50, 0.80)
        3-pt Avg:          0.1853
```

```
Fallout - Recall Averages (recall after X non-relevant docs retrieved):

    At   0 docs:   0.0578

    At  14 docs:   0.1821

    At  28 docs:   0.2415

    At  42 docs:   0.2806

    At  56 docs:   0.3257

    At  71 docs:   0.3635

    At  85 docs:   0.3759

    At  99 docs:   0.3910

    At 113 docs:   0.4337

    At 127 docs:   0.4531

    At 142 docs:   0.4615

Average recall for first 142 non-relevant docs retrieved:

                  0.3268
```

Appendix B: User Interface of Ai-Times



新闻标题	下载时间	新闻来源	新闻类别	图像
油市早评 2005-02-24	2005-01-25	广发期货	股票快讯	☐
继续微幅收跌-外棉早评2.25 2005-02-25	2005-01-25	广发期货	股票快讯	☐
获利了结打压铜价回落---LME铜早评 2005-02-25	2005-01-25	广发期货	股票快讯	☐
外盘带动 创出新高-郑棉日评2.23 2005-02-24	2005-01-25	广发期货	股票快讯	☐
关注出口报告 大涨之后小幅回调-外棉早评2.24 2005-02-24	2005-01-25	广发期货	股票快讯	☐
震荡整理　基金继续平仓　玉米早评 2005-02-25	2005-01-25	广发期货	股票快讯	☐
伦铜在高位整固---LME铜早评 2005-02-24	2005-01-25	广发期货	股票快讯	☐
四度空间看大豆：　连豆上冲受阻 2005-02-25	2005-01-25	广发期货	股票快讯	☐

油市早评 2005-02-24　(2005-01-25)　(金融总汇)

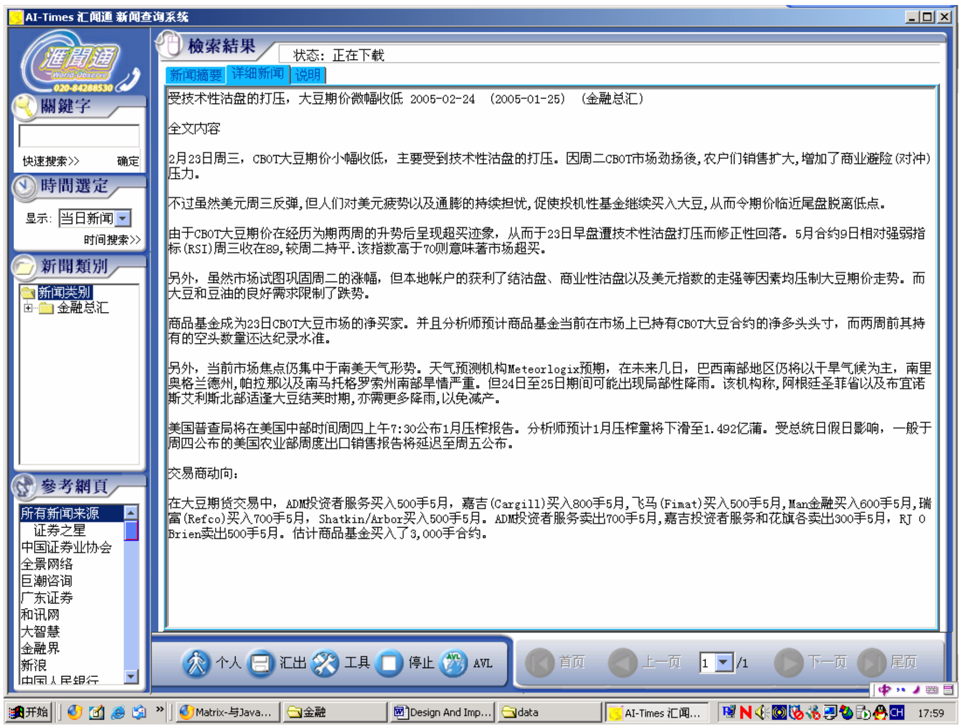
摘要内容

油市早评

纽约商业期货交易所(NYMEX)原油期货周三收低,此前石油输出国组织(OPEC)表示将不会允许油价过度上涨,从而削弱了油价周二近6%的涨幅.但取暖油合约触及13周高点,抢去了原油期货的风头,因市场预期美国在周四早间公布馏分油库存数据将减少.4月原油期约结算价下滑0.25美元,报每桶51.17,此前曾触及盘中高点51.60,盘中低点报50.88。一些交易商认为取暖油的涨势似乎有些过度,因供给充足,且冬季自目前还不足一个月就将结束

伦敦国际石油交易所(IPE)布兰特原油期货周三收低，4月布兰特原油期货结算价下滑0.11美元,报每桶48.51。

Fig. 27. User Interface of Ai-Times: Retrieval Result



受技术性沽盘的打压，大豆期价微幅收低 2005-02-24　(2005-01-25)　(金融总汇)

全文内容

2月23日周三，CBOT大豆期价小幅收低，主要受到技术性沽盘的打压。因周二CBOT市场劲扬後,农户们销售扩大,增加了商业避险(对冲)压力。

不过虽然美元周三反弹,但人们对美元疲势以及通膨的持续担忧,促使投机性基金继续买入大豆,从而令期价临近尾盘脱离低点。

由于CBOT大豆期价在经历为期两周的升势后呈现超买迹象，从而于23日早盘遭技术性沽盘打压而修正性回落。5月合约9日相对强弱指标(RSI)周三收在89,较周二持平.该指数高于70则意味著市场超买。

另外，虽然市场试图巩固周二的涨幅，但本地帐户的获利了结沽盘、商业性沽盘以及美元指数的走强等因素均压制大豆期价走势。而大豆和豆油的良好需求限制了跌势。

商品基金成为23日CBOT大豆市场的净买家。并且分析师预计商品基金当前在市场上已持有CBOT大豆合约的净多头头寸，而两周前其持有的空头数量还达纪录水准。

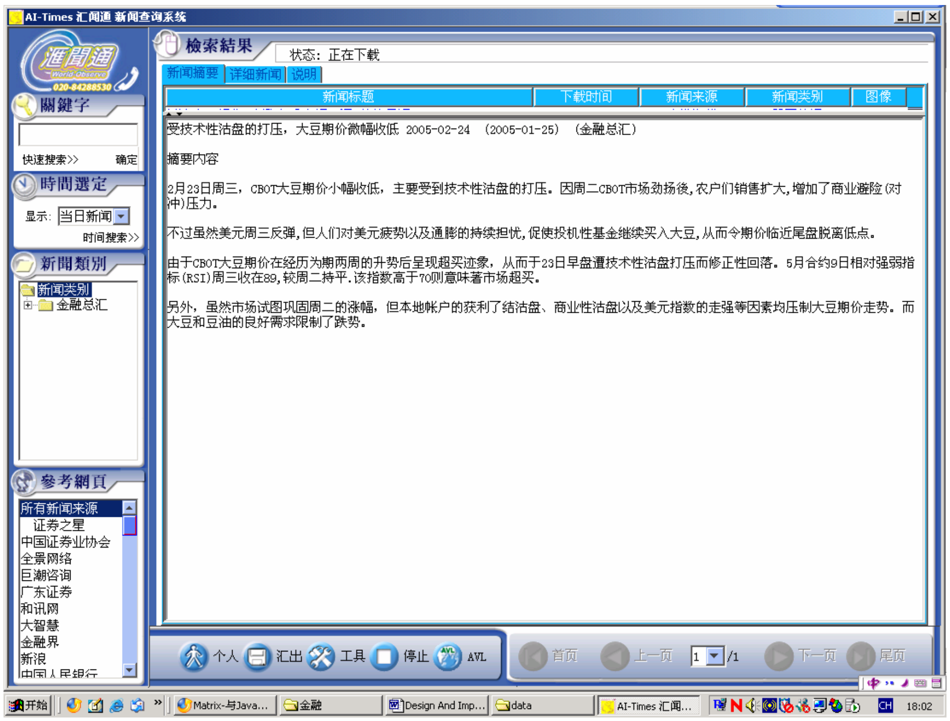
另外，当前市场焦点仍集中于南美天气形势。天气预测机构Meteorlogix预期，在未来几日，巴西南部地区仍将以干旱气候为主，南里奥格兰德州,帕拉那以及南马托格罗索州南部旱情严重。但24日至25日期间可能出现局部性降雨。该机构称,阿根廷圣菲省以及布宜诺斯艾利斯北部适逢大豆结荚时期,亦需更多降雨,以免减产。

美国普查局将在美国中部时间周四上午7:30公布1月压榨报告。分析师预计1月压榨量将下滑至1.492亿蒲。受总统日假日影响，一般于周四公布的美国农业部周度出口销售报告将延迟至周五公布。

交易商动向：

在大豆期货交易中，ADM投资者服务买入500手5月，嘉吉(Cargill)买入800手5月,飞马(Fimat)买入500手5月,Man金融买入600手5月,瑞富(Refco)买入700手5月，Shatkin/Arbor买入500手5月。ADM投资者服务卖出700手5月,嘉吉投资者服务和花旗各卖出300手5月，RJ O Brien卖出500手5月。估计商品基金买入了3,000手合约。

Fig. 28. User Interface of Ai-Times: News Content



受技术性沽盘的打压，大豆期价微幅收低 2005-02-24 （2005-01-25） （金融总汇）

摘要内容

2月23日周三，CBOT大豆期价小幅收低， 主要受到技术性沽盘的打压。因周二CBOT市场劲扬後,农户们销售扩大,增加了商业避险(对冲)压力。

不过虽然美元周三反弹,但人们对美元疲势以及通膨的持续担忧,促使投机性基金继续买入大豆,从而令期价临近尾盘脱离低点。

由于CBOT大豆期价在经历为期两周的升势后呈现超买迹象， 从而于23日早盘遭技术性沽盘打压而修正性回落。5月合约9日相对强弱指标(RSI)周三收在89,较周二持平.该指数高于70则意味著市场超买。

另外，虽然市场试图巩固周二的涨幅，但本地帐户的获利了结沽盘、商业性沽盘以及美元指数的走强等因素均压制大豆期价走势。而大豆和豆油的良好需求限制了跌势。

Fig. 29. User Interface of Ai-Times: News Summary