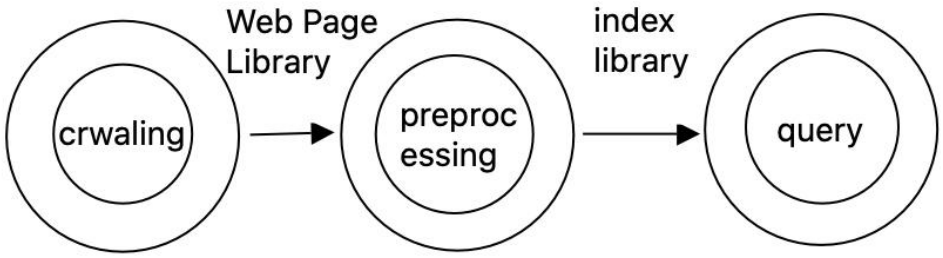
# System Structure of Search Engine(Coo)

## Abstract

This report includes three parts, and will explain step by step how to design and implement a search engine. In the first part, you will first learn the working principle of the search engine and understand its architecture, and then explain how to implement the first part of the search engine, the web crawler module, which completes the web page collection function. In the second part of the series, we will introduce the preprocessing module, that is, how to process the collected web pages, and organize, segmentation, and indexing are all in this part. In the third part of the series, the implementation of the information query service will be introduced, mainly the establishment of the query interface, the return of query results, and the implementation of snapshots.

### The overall structure of Coo and the process of data transmission.

In fact, the three parts of the search engine are independent of each other, and the three parts work separately. The main relationship is reflected in the data obtained in the previous part. The relationship between the three is shown in the following figure:
Figure 1. Search engine three-stage workflow




The top-down approach describes the search engine execution process:
The user submits the query word or phrase P through the browser, and the search engine returns a list of matching web page information (L) according to the user query;

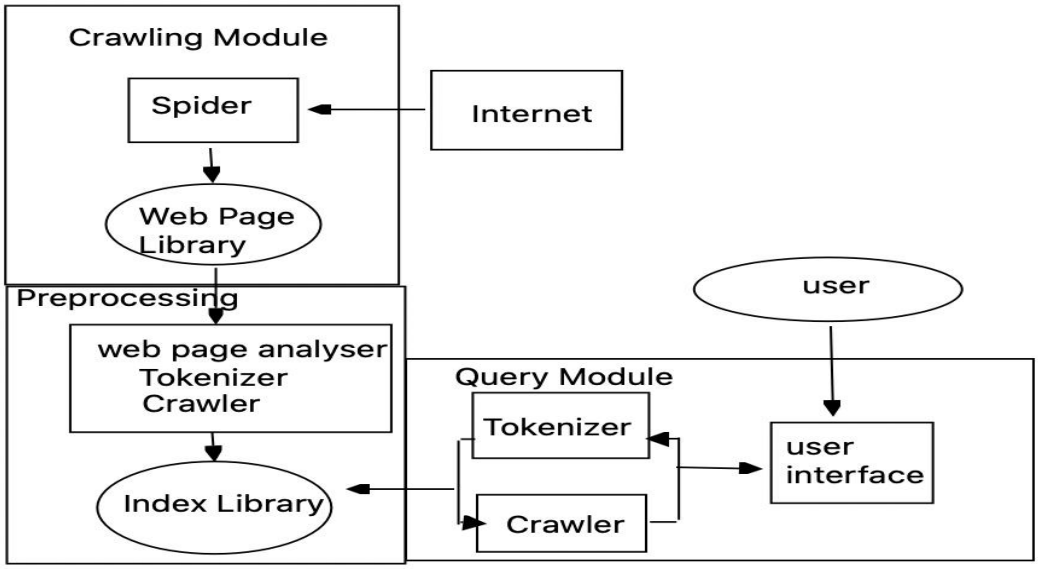
The above process involves two questions, how to match the user's query and where does the web page information list come from, and according to what sort? The user's query P is cut into small phrases <p1, p2… pn> by the tokenizer. According to an inverted index maintained by the system, a certain word pi can be queried in which web pages have appeared, matching those pages where <p1, p2… pn> appears can be used as the initial result. Furthermore, the returned initial web page set can calculate the relevance of the query term to obtain the page rank, that is, Page Rank , You can get the final list of pages according to the ranking order of the pages;

Assuming that the formulas for the tokenizer and page ranking are both established, where does the inverted index and the original page set come from? In the introduction of the previous data flow of the original web page set, it can be known that the crawler spider crawls the web pages and saves them locally, and the inverted index, that is, the

phrase-to-page mapping table is based on the forward index. The latter is the web page-to-phrase mapping table obtained after analyzing the content of the web page and segmenting the content, and the inverted index can be obtained by inverting the positive index;

What does web page analyzer do? Because the original web page collected by the crawler contains a lot of information, such as html forms and some spam information such as advertisements, the web page analysis removes this information, and extracts the text information in it as subsequent basic data.

After the above analysis, we can get the overall structure of the search engine as follows:




The crawler crawls a large number of web pages from the Internet and stores them as the original web page library. Then the web page analyzer extracts the topic content from the web page and sends it to the tokenizer for word segmentation. The obtained results use the indexer to establish forward and backward indexes. The index database is obtained. When the user queries, the input query phrase is cut by the tokenizer and the query is performed in the index database by the searcher. The obtained result is returned to the user.

No matter the size of the search engine, its main structure is composed of these parts, and there is no big difference. The quality of the search engine is mainly determined by the internal implementation of each part.

With the above-mentioned overall understanding of search engines, let's learn the specific design and implementation of the crawler module in Coo.

## The concrete implementation of Spider

### Web collector Gather

The web page collector obtains the web page data corresponding to the URL through a URL key word. Its implementation mainly uses the URLConnection class in Java to open the network connection of the corresponding page of the URL, and then

reads the data in the I / O stream. The buffer of data improves the efficiency of data reading and the readLine () line reading function.

### Web processing

The collected single web page needs to be processed in two different ways. One is to put it into the web page library as the raw data for subsequent processing; the other is to analyze the URL link and extract it into the URL pool to wait for the corresponding webpage. Web pages need to be saved in a certain format for batch processing of future data. The web page library consists of several records, each record contains a piece of web page data information, and the storage of records is added sequentially; The header is composed of several attributes, including: version number, date, IP address, data length, arranged according to the attribute name and attribute value, with a colon in the middle, and each attribute occupies a line; Data is web page data. It should be noted that the reason for adding the date of data collection is that since the content of many websites is dynamically changed, such as the homepage content of some large portals, this means that if it is not the web page data that is crawled that day, the data may be out of date Problem, so you need to add date information to identify it.

URL extraction is divided into two steps. The first step is URL identification. The second step is to organize the URL. The two steps are mainly because the links of some websites use relative paths. If they are not organized, errors will occur. URL recognition is mainly through regular expression matching. The process first sets a string as the matching string pattern, and then after compiling in the pattern, you can use the Matcher class to match the corresponding string. The implementation code is as follows:
According to String pattern1 = "(\\S)*(news.yahoo.com)[-A-Za-z0-9/]+[-A-Za-z0-9/_]";
this regular expression can match the entire tag where the URL is located, so after the loop gets the entire tag, you need to further extract the real URL, we can intercept the content between the first two quotes in the tag to get this content. After that, we can get a preliminary URL collection that belongs to the web page.

Next, we perform the second step, the URL sorting, that is, filtering and integrating the URL collection in the entire page obtained previously.Since we can easily obtain the URL of the current web page. On the other hand, among the comprehensive URLs included in the page, there are some web pages such as video or image web pages that we do not want to crawl or are not important. Here we mainly perform a simple processing for these elements on the page by adding a boolean express like page.hasText(). For example, when the connection contains expressions such as "jpg", the priority of the link can be lowered, so that crawling of advertisement links can be avoided to a certain extent.

### Dispatcher

The allocator manages URLs, which is responsible for saving the URL pool and distributing new URLs after Gather gets a certain web page, and also avoids repeated collection of web pages. The allocator is coded in the singleton pattern in the design pattern and is responsible for providing Gather with the new URL. The singleton pattern is particularly important because it involves subsequent multithreading rewrites.

Duplicate collection refers to a web page that physically exists and is repeatedly accessed by Gather without updating. This results in a waste of resources. The main reason is that there is no clear record of the URLs that have been accessed and cannot be discerned. Therefore, Dispatcher maintains two lists, "visited tables" and "unvisited tables". After the page corresponding to each URL is crawled, the URL is placed in the visited table, and the URL extracted from the page is placed in the unvisited table. When Gather requests the URL from Dispatcher, first verify whether the URL in the visited table.

Spider launches multiple Gather threads.Now that there are hundreds of millions of web pages in the Internet, it is obviously inefficient for a single Gather to collect web pages, so we need to use a multi-threaded method to improve efficiency. Gather's function is to collect web pages. We can use the Spider class to start multiple Gather threads to achieve the purpose of multi-threading. code show as below:

After the thread is started, the web collector starts the operation of the job, and after one job is completed, it applies to the Dispatcher for the next job. Because of the multi-threaded Gather, in order to avoid thread unsafeness, it is necessary to perform exclusive access to the Dispatcher. The synchronized keyword is added to its function to achieve thread-safe access.