

CPS842 Assignment 3: Vector Space Search Engine - Complete System

Jeffrey Mair

November 28, 2013

Contents

1.1	Overview	3
1.2	WebCrawler.jar	3
1.2.1	Worker Threads	3
1.2.2	WebCrawler.jar Usage	4
1.3	Database	5
1.3.1	Database Table Descriptions	7
1.3.2	Encoding & Collation Considerations	7
1.4	Web Services	7
1.4.1	GetPagesToCrawl.php	7
1.4.2	AddCrawlResult.php	7
1.4.3	AddDocumentTerms.php	8
1.4.4	GetUnverifiedPages.php	8
1.4.5	GetScoredPageIdsFromQuery.php	8
1.5	Web User Interface	8
1.5.1	Search Page	8
1.5.2	Stats Page	9
1.5.3	URL Submission Page	9
1.6	Possible Improvements	11
1.7	Resources	11

List of Figures

1.1	System Components	3
1.2	Starting WebCrawler.jar	5
1.3	Running WebCrawler.jar	5
1.4	Database Diagram	6
1.5	Fast Cosine Similarity calculation algorithm from Introduction to Information Retrieval, section 7.1	8
1.6	UI - Search Page	9
1.7	UI - Stats Page	10
1.8	UI - URL Submission Page	10

1.1 Overview

This document describes the creation of a custom web search engine based on the vector-space information retrieval model. The system was designed to be portable such that it can be run on standard low-cost web hosting that supports PHP/MYSQL. The components in the system include:

- Web Crawler & Indexer (Java)
- Database (MySQL)
- Web Services & User Interface (PHP)

Figure 1.1 shows a visual overview of the components.

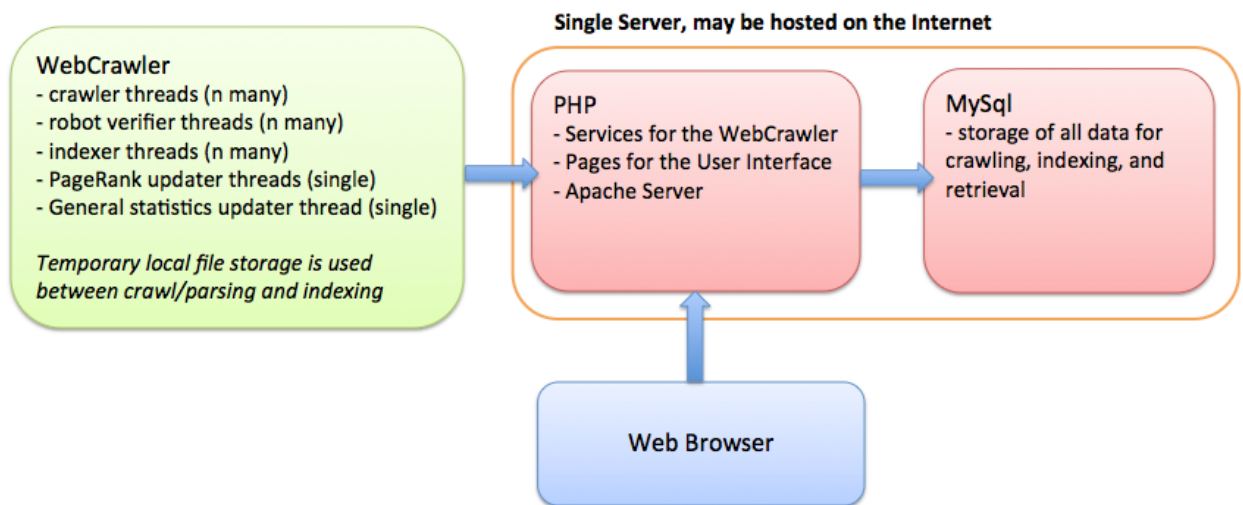


Figure 1.1: System Components

1.2 WebCrawler.jar

Both the Web Crawler and Indexer were implemented as a single Java program which shall be referred to simply as WebCrawler.jar. The program is multi-threaded to allow for multiple operations to occur simultaneously, as some operations are more time-consuming than others, and it is ideal to not have any components idly waiting for their turn to run.

1.2.1 Worker Threads

There are multiple threads running within WebCrawler.jar, and they are described in the following subsections.

Crawler Threads (multiple)

The crawler thread requests a set of verified URL's from the database and begins to download the pages one at a time. As each page is downloaded, it is parsed for new hyperlinks which are added to the database and flagged as "unverified". This flag prevents the crawler thread from downloading those pages until the Robot Verifier runs and ensures that they are appropriate to download. After the Crawler

parses each page, they are saved to the disk, with the HTML tags removed, in a local folder where the indexer threads can retrieve them. The file is prefixed with the name `crawler#` so that the corresponding Content Indexer threads know which files they should index.

Each crawler is assigned an ID which is associated with the domains of any URLs it finds. That is used to ensure that crawlers do not try to crawl the same domains as each other as there may be a user-defined number of crawler threads running in the program.

Robot Verifier Threads (multiple)

For each crawler thread, there is a corresponding Robot Verifier thread whose responsibility is to verify each URL found by the crawler. The Robot Verifier checks for a `Robots.txt` file at the domain of each unverified URL that has been stored in the database against a particular crawler ID. If the `Robots.txt` file indicates that the URL should not be indexed, it is deleted from the database, otherwise it is marked as verified.

Content Indexer Threads (multiple)

There is a Content Indexer thread for each crawler thread. The Content Indexer continually checks the folder that the Crawler thread deposits web page content files into. The Content Indexer only reads in files that begin with `crawler#`, where the `#` corresponds to their ID, which in turn matches the crawler that deposited the file there. This ensures that Indexers are not trying to index the same file.

Page Rank Calculator Thread (single)

The PageRank calculations are updated periodically by a single thread that is spawned by the first (`#1`) Content Indexer thread. The interval of which this PR updating thread is created/started is user-defined (see `WebCrawler.jar Usage` section).

Statistics Updater Thread (single)

Similar to the Page Rank Calculator Thread, another thread is periodically created at the same time which updates any database statistics that need to be updated. As of this final release of the software, only the total number (`'N'`) of indexed pages needs to be updated. This value is used by the cosine-similarity calculation during the scoring of pages against a query. This `N` value is updated outside of the scoring to improve performance.

1.2.2 WebCrawler.jar Usage

The `WebCrawler.jar` has several input arguments that determine its behaviour. They are listed here:

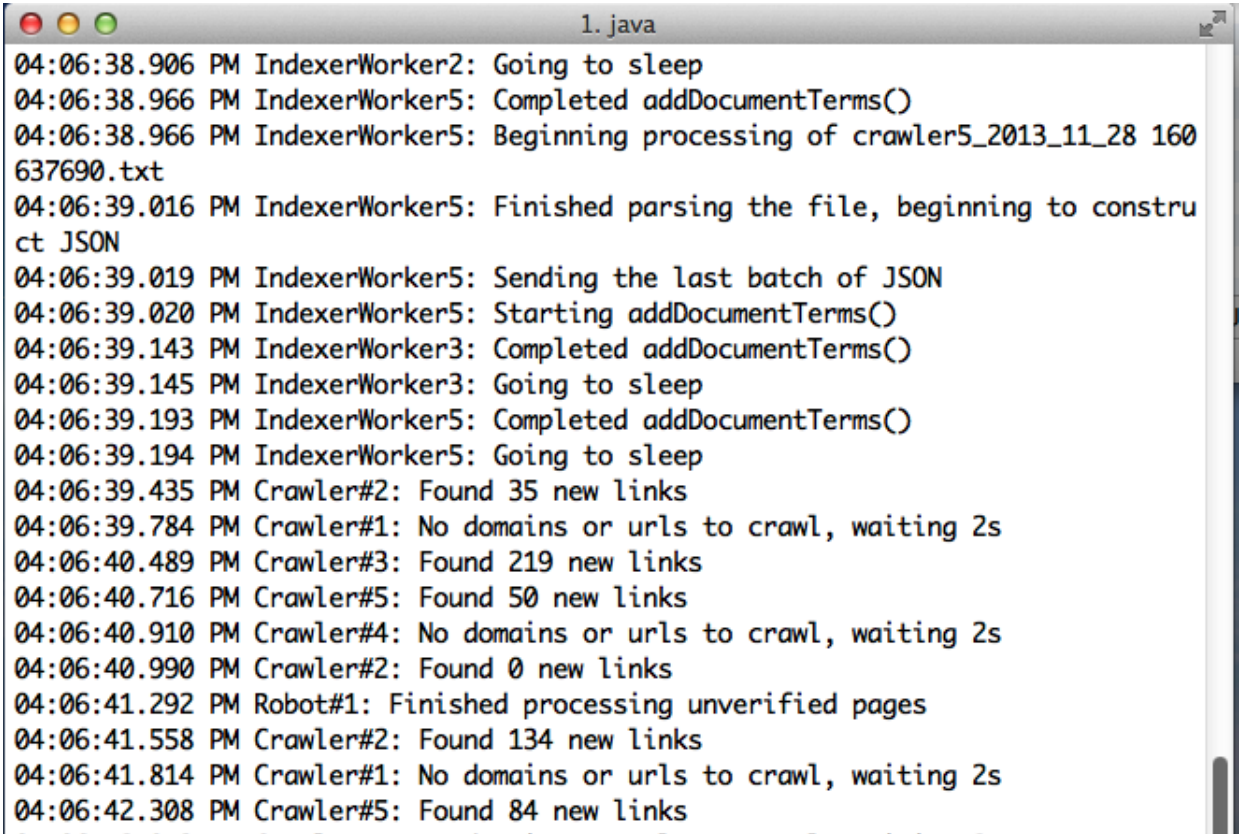
- - **-crawl={true/false}**. This tells the program to run or not run any crawler threads.
- - **-index={true/false}**. This tells the program to run or not run any content indexer threads.
- - **-host={computer-name}**. This is the name of the computer where the PHP web services are located that `WebCrawler.jar` will use to request and submit data. So this may be `localhost` for locally configured systems, or a website such as `mywebsite.com`. (The `WebCrawler` has hard-coded web service paths following the host name; more detail on that in the Web Services section).

- - **-printerval={integer}**. This is the number of documents that should be indexed before re-running PageRank updating (the Page Rank Calculator Thread).
- - **-workers={integer}**. This is the number of Crawler, Robot Verifier, and Content Indexer threads to create. You may set this based on your available resources on the computer that is hosting WebCrawler.jar. For example, if set to 5, there will be 5 crawlers, 5 robot verifiers, and 5 indexers. Each will be dedicated to their own specific domains.

Figure 1.2 shows an example of starting WebCrawler.jar, and figure 1.3 shows typical output sent to the terminal while running.

```
Jeffs-Mac-mini:WebCrawler Jeff$ java -jar -Xmx2048m WebCrawler.jar --crawl=true
--index=true --host=localhost --printerval=100 --workers=5
```

Figure 1.2: Starting WebCrawler.jar



```
04:06:38.906 PM IndexerWorker2: Going to sleep
04:06:38.966 PM IndexerWorker5: Completed addDocumentTerms()
04:06:38.966 PM IndexerWorker5: Beginning processing of crawler5_2013_11_28 160
637690.txt
04:06:39.016 PM IndexerWorker5: Finished parsing the file, beginning to constru
ct JSON
04:06:39.019 PM IndexerWorker5: Sending the last batch of JSON
04:06:39.020 PM IndexerWorker5: Starting addDocumentTerms()
04:06:39.143 PM IndexerWorker3: Completed addDocumentTerms()
04:06:39.145 PM IndexerWorker3: Going to sleep
04:06:39.193 PM IndexerWorker5: Completed addDocumentTerms()
04:06:39.194 PM IndexerWorker5: Going to sleep
04:06:39.435 PM Crawler#2: Found 35 new links
04:06:39.784 PM Crawler#1: No domains or urls to crawl, waiting 2s
04:06:40.489 PM Crawler#3: Found 219 new links
04:06:40.716 PM Crawler#5: Found 50 new links
04:06:40.910 PM Crawler#4: No domains or urls to crawl, waiting 2s
04:06:40.990 PM Crawler#2: Found 0 new links
04:06:41.292 PM Robot#1: Finished processing unverified pages
04:06:41.558 PM Crawler#2: Found 134 new links
04:06:41.814 PM Crawler#1: No domains or urls to crawl, waiting 2s
04:06:42.308 PM Crawler#5: Found 84 new links
```

Figure 1.3: Running WebCrawler.jar

1.3 Database

A MySQL database was designed to house the information related to web crawling, as well as all the indexed pages and terms. The only data that is not persisted to this database is the downloaded webpages that are waiting to be indexed by the Content Indexers. A diagram of the database is shown in figure 1.4.

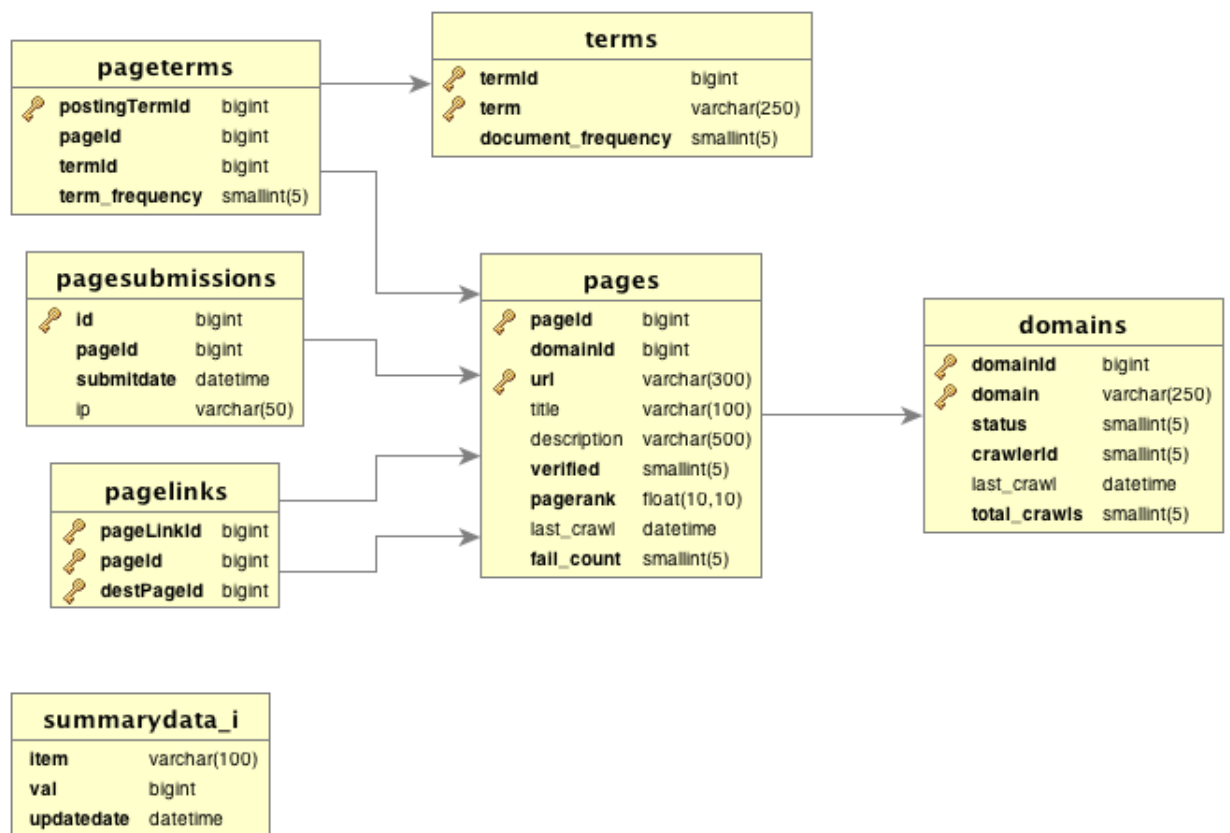


Figure 1.4: Database Diagram

1.3.1 Database Table Descriptions

Here is a brief description of the purpose of each table (and most significant columns) in the database:

- **domains:** Contains a list of domains for each page that has been found. The crawlerId dedicates specific domains to specific crawlers so there is no overlap. Last_crawl is used to prevent crawling domains too frequently.
- **pages:** Contains a list of pages. Verified indicates if the Robot Verifier has ensured that the page can be crawled. Title and Description are optional and inserted after crawling if they are found in the web page. Last_crawl and fail_count are used to adjust the priority of crawling the page.
- **pagelinks:** Contains the links found in each page. This constitutes the web graph and is used for calculating PageRank.
- **pagesubmissions:** This table keeps a record of pages submitted via the SubmitSite webpage. It tracks the user's IP address and time of submission.
- **terms:** This table contains all the distinct terms that have been cataloged by the indexers. Included is the Document Frequency which is updated during indexing.
- **page terms:** Pageterms has the instances of terms used in each page, and the Term Frequency.
- **summarydata.i:** This table is essentially for key-value pair storage of any extraneous data such as total pages indexed.

1.3.2 Encoding & Collation Considerations

In order to support languages with non-Latin characters, the fields that contain 'content' from the web (terms, title, description) needed to be configured to use utf8 character encoding, and 'COLLATE utf8_bin' collation.

1.4 Web Services

The WebCrawler.jar program interacts with the Database by exchanging JSON-formatted data via PHP web services. There are numerous services in this project, so only some very pertinent ones are listed in the following sections:

1.4.1 GetPagesToCrawl.php

The Crawler threads get a list of pages to crawl (to populate their URL frontier) using this service. The service contains logic to prevent crawling the same domain or page too frequently. The rules are:

- Page verified == 1 (must be verified by the robot verifier)
- and, page.domain.crawler id must match the requesting crawler
- and, domain must not have been crawled in

1.4.2 AddCrawlResult.php

The Crawler threads submit crawl results using this service.

1.4.3 AddDocumentTerms.php

The Content Indexer threads submit page terms using this service.

1.4.4 GetUnverifiedPages.php

The Robot Verifier threads get their list of page to verify from this service. This includes pages marked as unverified which are from a domain associated with this Robot Verifier's corresponding Crawler.

1.4.5 GetScoredPageIdsFromQuery.php

This service is used by the User Interface to query the database and returned ranked pages. In this service, the “fast” cosine similarity algorithm from Section 7.1 of Introduction to Information Retrieval, (Manning, Raghavan, and Schutze) was implemented. This simplified top cosine similarity formula eliminates the need to work with very large sparse vectors. Since most queries consist of only a few terms (far less than the number of terms in each document), working with full document vectors is highly inefficient. This algorithm finds the relative scores of the documents, not the absolute scores. The algorithm is shown in figure 1.5.

```
FASTCOSINESCORE(q)
1  float Scores[N] = 0
2  for each d
3  do Initialize Length[d] to the length of doc d
4  for each query term t
5  do calculate  $w_{t,q}$  and fetch postings list for t
6    for each pair(d,  $tf_{t,d}$ ) in postings list
7    do add  $wf_{t,d}$  to Scores[d]
8  Read the array Length[d]
9  for each d
10 do Divide Scores[d] by Length[d]
11 return Top K components of Scores[]
```

Figure 1.5: Fast Cosine Similarity calculation algorithm from Introduction to Information Retrieval, section 7.1

1.5 Web User Interface

The Web UI was implemented using HTML, CSS, PHP, and Javascript.

1.5.1 Search Page

The Search page has only a text box for submitting a multi-word user query, and the navigation links. After the user types their query and presses Enter, the results are requested from the appropriate PHP function using a javascript AJAX call. AJAX was used so that the page does not disappear and re-appear which would be visually disruptive. Figure 1.6 shows an example of search results in the page.

The results on the search page are ranked in order from highest to lowest score. The score is shown in the top-right corner of the shown item, and the PageRank is shown just below the score. Also shown

in each result is a hyperlink to the page (where the hyperlink text is the page title if it was found by the crawler), and the URL below the hyperlink, as well as the page description (if available).

The duration of the query is also shown near the upper-right part of the results.

[Search](#) [Stats](#) [Submit Site](#)

retrieval

Results:

Query time: 1s

Center for Intelligent Information Retrieval http://ciir.cs.umass.edu/	Score: 0.0307075 PageRank: 0.0000000
Introduction to Information Retrieval http://nlp.stanford.edu/IR-book/	Score: 0.0232012 PageRank: 0.0000000
Web search basics http://nlp.stanford.edu/IR-book/html/htmledition/web-search-basics-1.html Web search basics	Score: 0.0225223 PageRank: 0.0000000
http://nlp.stanford.edu/IR-book/html/htmledition/dictionaries-and-tolerant-retrieval-1.html http://nlp.stanford.edu/IR-book/html/htmledition/dictionaries-and-tolerant-retrieval-1.html	Score: 0.0191101 PageRank: 0.0000000
Jefftron I anything i like http://jefftron.com/	Score: 0.0166157 PageRank: 0.0006799
Hierarchical clustering http://nlp.stanford.edu/IR-book/html/htmledition/hierarchical-clustering-1.html Hierarchical clustering	Score: 0.0132484 PageRank: 0.0000000

Figure 1.6: UI - Search Page

1.5.2 Stats Page

The Stats page shows some of the significant metrics about the system including total pages indexed, total distinct terms, posting terms, and database size. Also shown is the top 25 Page Rank scores. See figure 1.7.

1.5.3 URL Submission Page

The URL submission page takes in a web page URL and adds it to the list of pages to be crawled after robot verification. A single crawler/robot/indexer is allocated to watching for user-submitted pages so that they do not take a long time before being indexed.

Upon submission of a URL, the user's IP address is also stored in case the input URL is inappropriate.

Search Engine Stats

Total pages indexed: 1271 (2013-11-26 10:04:42)

Total distinct terms: 45267

Total posting terms: 374528

Database Size (in MB): 204

Top 25 Page Rank scores

Pos	PageRank	Site
1	0.0013268150	http://www.nhncorp.com/
2	0.0013242377	Y Combinator: Startup Library
3	0.0010754087	Here's who (probably) did that massive \$150,000,000 Bitcoin transaction
4	0.0010754087	Y Combinator: Winter 2014 Funding
5	0.0010615769	Paul Gribble
6	0.0010562448	5 Useful CSS Tricks for Responsive Design
7	0.0010562448	http://library.ryerson.ca/info/hours/
8	0.0010438034	Ferro Donations
9	0.0010166311	The Purdue Writing Lab - Welcome to the Writing Lab at Purdue
10	0.0010166311	http://owl.english.purdue.edu/owl/
11	0.0009411314	http://jefftron.com/school/
12	0.0009347926	한게임 - 즐거운 쉼표, Go 한게임!
13	0.0009246494	http://www.ryerson.ca/accessibility/

Figure 1.7: UI - Stats Page

Enter a page url to index:

Note: indexing may not be turned on currently, so your URL may not be indexed very soon as this Search Engine is currently experimental

Figure 1.8: UI - URL Submission Page

1.6 Possible Improvements

Given the relatively short design and construction period for this system, there are some areas that might benefit from further attention for making improvements. The following sections list some such areas.

Caching

Since many users will often make the same requests as each other, it would make sense to cache the scores for specific popular queries. After a query has been scored, the results can be stored in the database for immediate retrieval during the next request for the very same query. After some reasonable amount of time, the scores may be either discarded or updated depending on whether users are still frequently using the same query.

Security

Due to the time constraints on this project, only minimal security was implemented in the PHP services. Further effort to improve the robustness of the security would be advisable.

1.7 Resources

Some freely available libraries were used in the development of this system. They include:

- JSoup for HTML parsing (<http://jsoup.org/>)
- Jung for PageRank calculation (<http://jung.sourceforge.net/site/apidocs/edu/uci/ics/jung/algorithms/scoring/PageRank.html>)
- Quick Json for json decoding (<https://code.google.com/p/quick-json/>)

Bibliography

- [1] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schutze, *Introduction to Information Retrieval*. Cambridge University Press. 32 Avenue of the Americas, New York, NY 10013-2473, USA. 2008.