**BIG DATA TERM PROJECT**

3460:689-401

Instructor: Dr. C. -C. Chan

**WEB CRAWLER**

**Apache Nutch**

Divya Meka Bhoomika Patel Kadivar

Department of Computer Science Department of Computer Science

Email: [dm182@zips.uakron.edu Em](mailto:sc175@zips.uakron.edu)ail: [bpk20@zips.uakron.edu](mailto:yp14@zips.uakron.edu)

Jaswanth Datt Meka

Department of Computer Science

Email: [jm212@zips.uakron.edu](mailto:jm212@zips.uakron.edu)

**Contents**

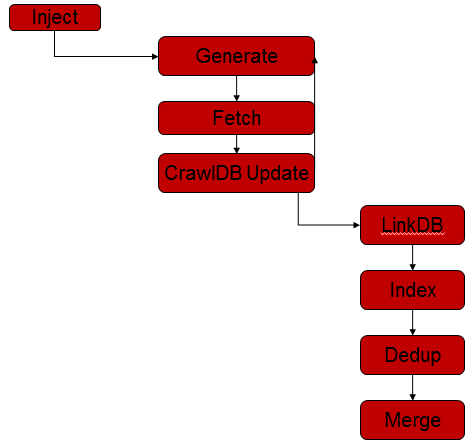
|  |  |  |
| --- | --- | --- |
| **1.** | **Introduction--------------------------------------------------------------------** | **3** |
| **2.** | **Web Crawler-------------------------------------------------------------------** | **3** |
| **3.** | **Apache Nutch------------------------------------------------------------------** | **3** |
|  | **3.1. Nutch Data----------------------------------------------------------------** | **4** |
| **4.** | **URL Seed List-----------------------------------------------------------------** | **4** |
| **5.** | **Robots.txt-----------------------------------------------------------------------** | **4** |
| **6.** | **Solr and Tika------------------------------------------------------------------** | **5** |
| **7.** | **Experiment---------------------------------------------------------------------** | **6** |
| **8.** | **Statistics-------------------------------------------------------------------------** | **6** |
| **9.** | **Conclusions-----------------------------------------------------------------------** | **9** |
| **10.** | **Acknowledgement---------------------------------------------------------------** | **9** |
| **11.** | **Remarks---------------------------------------------------------------------------** | **9** |
| **12.** | **References-------------------------------------------------------------------------** | **9** |

**ABSTRACT**

*A web crawler is an automated program that scans through internet webpages to collect information from the internet. Search engines use web crawlers so that when a search item is entered, crawler can quickly provide relevant web sites as a result. The data is growing every day and new websites are added to the already huge internet and web crawler ensures that databases are up-to-date and are updated as and when the online data changes. Web crawlers can be run periodically like in the case of search engines.*

1. **INTRODUCTION** In today’s world, users search everything on the web but before a search engine can give you the result let’s get into the details of how a search engine works. In order to retrieve a web page or a document, a search engine needs to find the document first. And to find information from millions of web pages on the web, a search engines employs a special software tool or application called spider to form lists of words to be found on web sites. The process of forming the lists is known as web crawling. Another important aspect of web crawling is web indexing; the contents of the web are indexed so that search engines can easily find all the relevant web pages using these indexes. Crawlers can copy all the pages they visit for later processing by a search engine.
2. **WEB CRAWLER**

The process of web crawling starts with a URL seeds list which is maintained in a queue. Every URL in the seeds list is accessed and all the hyperlinks scanned in the URL are scanned and the process continues until there is no more unvisited URL. The same steps are carried for all the URL’s in the seed list. The life cycle of a web crawler is shown in fig. 1 and each step is explained in the next section where Apache Nutch is used to crawl the web.



**Fig 2.1 Crawl Life Cycle**

1. **APACHE NUTCH**

Apache Nutch is a well matured production ready web crawler. It runs on top of Hadoop. It is scalable as it can process billions of pages, customizable as external plugin can be enabled to improve the functionality; it also provides extensible interfaces such as Parse, Index and Scoring Filter’s for custom applications. As seen in Fig. 1, the first step in the process of web crawling is to initialize Crawl DB and inject seed URL’s. We can decide how many cycles we want to run the web crawler and so repeat the generate-fetch-update cycle n times:

1. Generate fetch list: select URLs from
2. Crawl DB for fetching
3. Fetch URLs from fetch list
4. Parse documents: extract content, metadata and links
5. Update Crawl DB status, score and signature, add new URLs inlined or at the end of one crawler run (once for multiple cycles)
6. Invert links: map anchor texts to documents the links point to
7. Calculate link rank on web graph, update Crawl DB scores
8. Deduplicate documents by signature 8. Index document content, Meta data, and anchor texts
   1. **Nutch Data**

Nutch data is composed of:

* The crawl database, or crawl db. This contains information about every URL known to Nutch, including whether it was fetched, and, if so, when.
* The link database, or link db. This contains the list of known links to each URL, including both the source URL and anchor text of the link.
* A set of segments. A segment is generated for each cycle. Each segment is a set of URLs that are fetched as a unit. Segments are directories with the following subdirectories:
* a *crawl\_generate* names a set of URLs to be fetched
* a *crawl\_fetch* contains the status of fetching each URL
* a content contains the raw content retrieved from each URL
* a parse\_text contains the parsed text of each URL
* a parse\_data contains outlinks and metadata parsed from each URL
* a crawl\_parse contains the outlink URLs, used to update the crawldb

1. **URL SEED LIST**

We started our seedlist with Craigslist but were unable to crawl it due to the terms and conditions mentioned in the website which said “You agree not to use or provide software or services that interact or interoperate with client eg: for downloading, uploading, posting, flagging, emailing, search or mobile use. Robots, Spiders, crawlers, script, scrapers etc., are prohibited, as are misleading, unsolicited, unlawful, and/or spam posting/email. You agree not to collect user’s personal and/or contact information”. Due to this, we have considered [https://play.google.com/store](https://play.google.com/) as our seedlist. Every website has policies for handling crawlers. A file called robots.txt should be considered before crawling a website as it contains all the details about the contents of the website that can be crawled and the contents which should not be crawled.

1. **ROBOTS.TXT**

The robots.txt file contains permissions on which contents of the website can be accessed and crawled. The figures below show the robots.txt file for craigslist and for Google play store.



**Fig. 5.1 Craigslist robots file**



**Fig 5.2. Google Play store robots file**

As seen in the figures above, craigslist doesn’t provide any access to the contents of the website to crawlers whereas Google Play store allows few of its content to be crawled and remaining content is restricted.

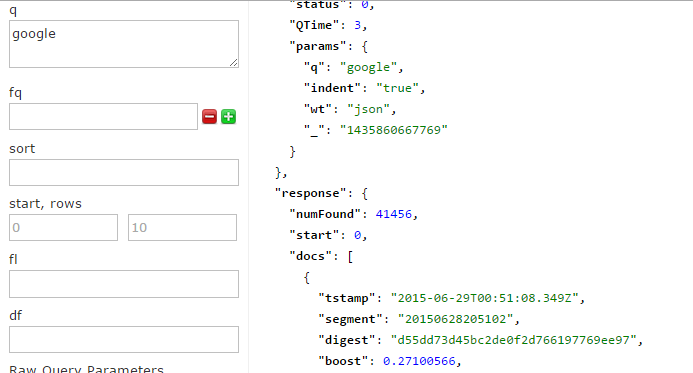
* If someone tries to access the restricted content of a website, law suits are made against the crawling organization. Some of these law suits are; Facebook lawsuit against API Harvester, Craigslist sues Pad Mapper that plots apartment listings on a map from Craigslist data, Baidu’s $15 Million Lawsuit against Qihoo 360.

1. **SOLR AND TIKA**

We have used Apache Solr 5.2.1 to integrate with our crawl database. Solr can be used to query our crawl DB database. By querying, we have found the number of occurrence of a particular word in our documents. Nutch Tika is used to parse different file formats like PDF, xlsx, doc/docx, txt, rss, rtf, swf etc. Also, Tika generates metadata about the documents which contains different properties of the document like author’s name, number of pages, file size etc. The metadata generated differs with the file type. Apache Tika and Solr are used to crawl, index and search documents.



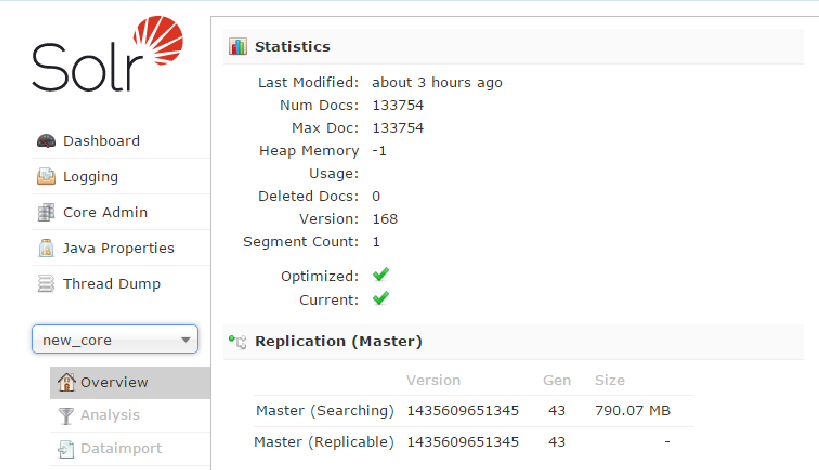
**Fig: 6.1. Querying for the word “google” in all our documents using Solr syntax in url.**



**Fig 6.2. Querying for the word “google” in all the documents using Solr interface.**

1. **EXPERIMENT**

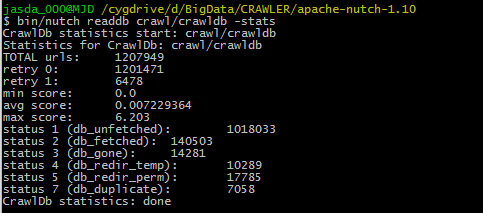
We ran the nutch crawler on Google play Store website for 22 hours and collected 790.07 MB of data which contains 133754 documents. We set the number of cycles to 9 and the topN value is set to 10000.The system configuration for our experimental setup is local system with – 12GB RAM, 750GB HDD, Intel i7 processor. We have used Apache Nutch 1.10 version and Solr 5.2.1 version.



**Fig: 7.1 Statistics of data crawled**

1. **STATISTICS**

The below figure shows the URL stats of the crawl performed.



**Fig: 8.1 Overview of CrawlDB statistics**

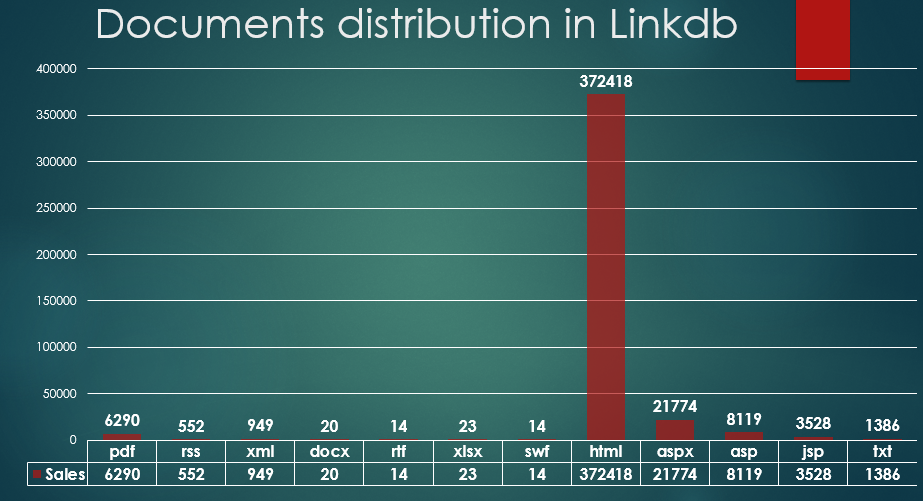
As seen in the figure, the total number of urls fetched is 1207949 out of which only 140503 URL’s are actually fetched. This is because in each cycle only the topN i.e. 10000 url’s go into the next cycle and all the remaining url’s even though are fetched doesn’t go to the next cycle and hence come under db\_unfetched url’s. The other reason for this can also be the filters and the unavailability of the URL temporarily. Db\_gone represents the number of urls that have a page not found error or a permanent error. Db\_redir\_temp is the number of urls that are temporarily redirected to another url before loading the actual url. Db\_redir\_perm represents the number of urls that are permanently redirected to a different url. Db\_duplicate represents the number of url that are repeated in the cycle and have duplicates presented in the seed list.

Every url is given a page rank to know the importance of the url over the web. This is done using OPIC calculation by considering the inlinks and outlinks as 1. In our experiment, we have found that [http: //gmpg.org/xfn/11](http://gmpg.org/xfn/11) has the highest score of 6.203019.

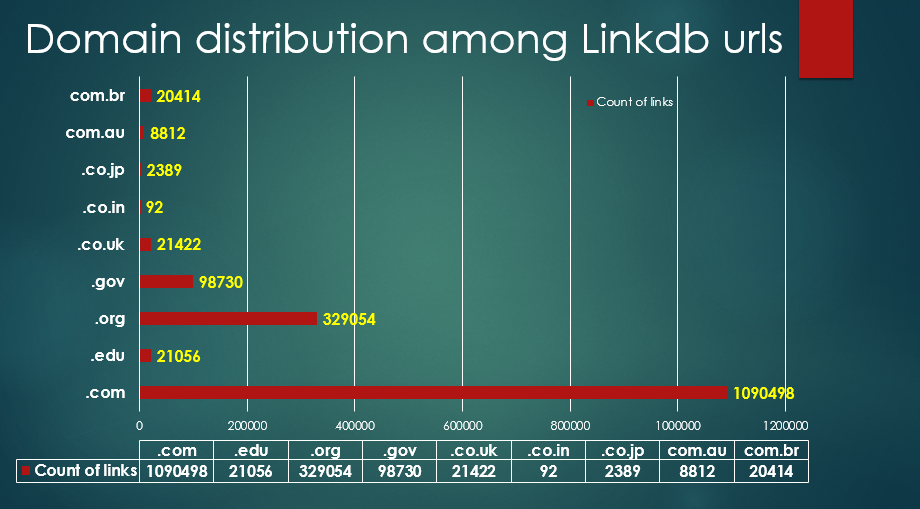


**Fig: 8.2. List of URL’s with scores in sorted order**

We have enabled Tika plugin in nutch-site.xml. This plugin helps to parse all documents with the following extension .pdf,.xlsx,.doc/docx, rss, and rtf. From all the parsed all URL links in Linkdb using Figure 8.3, we found that files with .html we parsed maximum times accounting 90% of all URL’s parsed. There are 6290 .pdf, 522 .rss URL’s that were parsed.

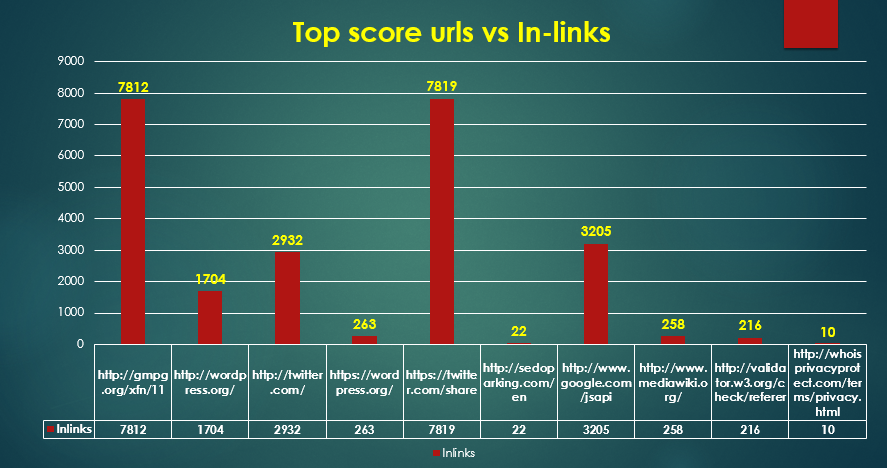
** Fig: 8.3. Distibution of Documents in LinkDB**

We have analyzed the distribution of domains among the URL’s that have been collected in Link DB. As observed in Figure 8.4, we have found that most of the URL’s belonged to .com domain followed by .org and then .gov from the URL list generated from our seedlist.



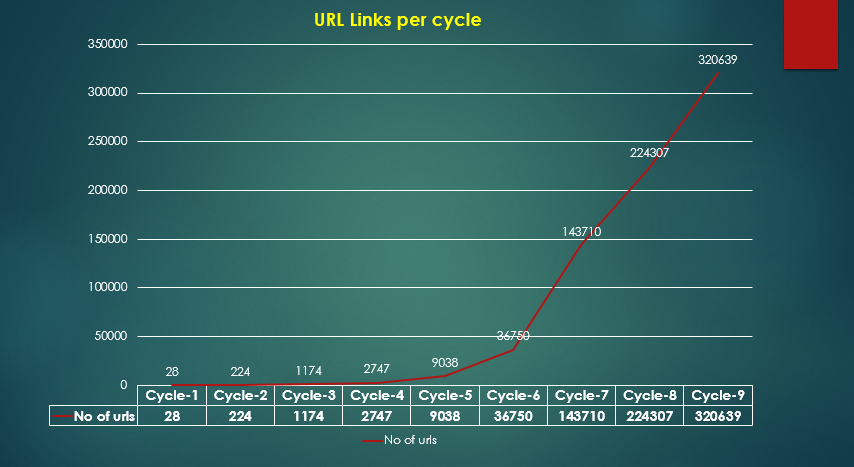
**Fig: 8.4. Distibution of Domain’s among URL list**

Continuing with our analysis,we wanted to check the Inlinks for the URL’s with Top scores calculated by default OPIC scoring.Using figure 8.5, we conclude that the highest scoring URL <http://gmpg.org/xfn/11>has highest number Inlinks.



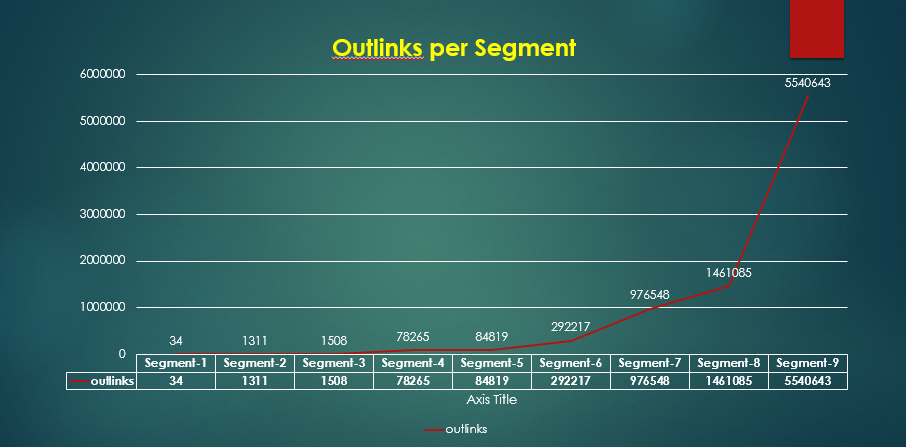
**Fig: 8.5. Top 10 scoring URL’s Vs InLinks**

We also wanted to analyze the segment level statistics for that data generated and stored in CrawlDb.We have plotted the URL links growth from segment to segment.As expected it conformed to our prediction that as cycles grow the growth in URL links per segment grows exponentially. Figure 8.6 confirms this trend.



**Fig .8.6 URL Links Vs Cycle(depth)**

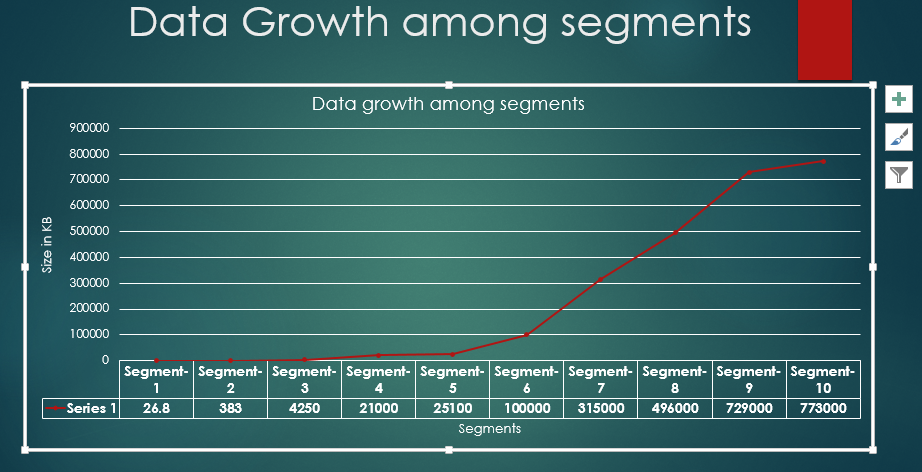
Looking further into the segment level statistics,we plotted URL outlinks per segment.



**Fig .8.7 URL Links Vs Cycle (depth)**

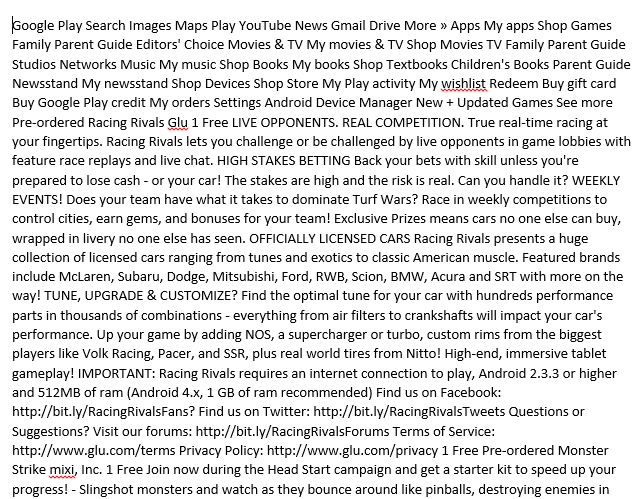
We observe the trend that as the number of cycles increases,the number of outlinks per segment increases exponentially.In the first cycle for [https://play.google.com/store](https://play.google.com/) there are 34 outlinks.

We also observed the trend of data growth among various segments. As the cycles progress, we see that data growth increases exponentially which is justified as the number of URL’s also has grown in the same way. Figure 8.8 confirms this trend.



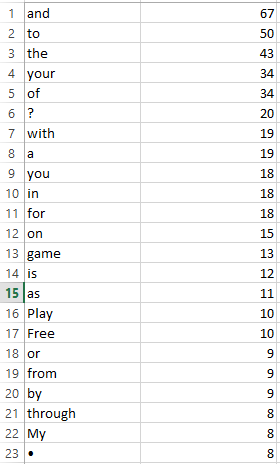
**Fig .8.8 Size in KB vs Segments**

From the data that we collected in our Crawldb, we wanted to parse data. We have parsed data for <https://play.google.com/store>.



**Fig: 8.9 Parsed text from google play store URL**

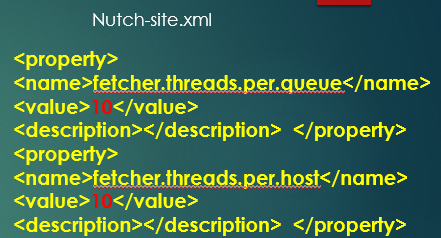
We have analyzed the distribution of words among this parsed text. We ran Map-Reduce program in CentosOS, to find the frequency distribution of words in the parsed text file. Figure 8.10 shows the distribution of words from the parsed text. ”and” was the most frequent word in our parsed text.



**Fig: 8.10 Frequency of words distribution.**

**9 CONCLUSIONS & FUTURE WORK**

From the data crawled and URL’s parsed, we have observed that there is very low fetch rate. The number of URL’s that have been unfetched has been significantly high. By increasing the number of inlinks collected and expanding the seed list, we plan to lower the unfetched rate significantly. Also, to improve the nutch performance, it is suggested to increase the number of threads that collect URL’s from queue increased to higher number (say 10) from default value of 1.



**Fig: 9.1 Improving Nutch Performance**

**10 ACKNOWLEDGMENT**

We sincerely thank Dr. Chan, Professor, Department of Computer Science, The University of Akron, for his valuable suggestions during the analysis and implementation phase of the project.

**11 STATEMENT OF WORK**

**Nutch Setup**: Divya Meka

Bhoomika P Kadivar

**Crawling**: Divya Meka

Bhoomika P Kadivar

**Analysis and**

**Reporting**: Jaswanth Datt Meka

**12 REFERENCES**

[1]

<http://events.linuxfoundation.org/sites/events/files/slides/aceu2014-snagel-web-crawling-nutch.pdf>

[2]

<https://en.wikipedia.org/wiki/Web_crawler>

[3]

<http://computer.howstuffworks.com/internet/basics/search-engine1.htm>

[4]

<https://wiki.apache.org/nutch/NutchTutorial>

[5]

<http://springboard.uakron.edu/d2l/le/content/4007326/Home?itemIdentifier=D2L.LE.Content.ContentObject.ModuleCO-3142424>

[6]

<http://www10.org/cdrom/papers/210/>