CSCI 572 HW3

COMPARING SEARCH ENGINE RANK ALGORITHMS

PRADEEP KUMAR NANDA KUMAR

USC ID: 9834726643

DATED: 29-Mar-2016

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| S.NO | Title | Page Number |
| 1 | Introduction and Objective | 3 |
| 2 | Setup and Installation | 4 |
| 3 | Working Principle | 5-10 |
| 4 | Analysis of Relevancy Results | 10-13 |

**Solutions as per grading guidelines and my submission**

1. Relevancy graphs that include Bing, Google,Solr Ranking, Pagerank Ranking Page 11 and 13.
2. Convincing explanation of the relevancy results - Page 12 and 13
3. Explanation regarding why some pages have higher pagerank values- Page 13
4. Screenshots of whole flow from query page till the result page for both ranking methods, and a screenshot showing one of the actual web documents which will open when we click on that result link. – Page 7, 8, 9, and 10
5. Correct structure of the external\_PageRankFile.txt . Attached

**FILES ATTACHED**

1. MyStatistics.xls and MyQueries.xls
2. QueryProcessor.php and QueryRequest.htm
3. Manage-schema and solrconfig.xml
4. External\_pageRank.txt and external\_pageRank\_Complete.txt
5. ScreenShots

REFERENCES:

1. <http://www-scf.usc.edu/~csci572/Slides/PageRank.pdf> - Lecture 12 csci 572 Spring 2016

2. Wikipedia – General Overview of Indexing and Page Ranking

**INTRODUCTION AND OBJECTIVE:**

Web Search engines like Google, Bing, Yahoo, etc use search algorithms to returns seemingly most wanted pages from a collection of web pages stored in their search space (database as records or any other data storage format).

“PageRank” algorithm is the algorithm used by Google to set the rank (score) based upon the importance of web site pages. A network of pages with incoming (citations) and outgoing (cites) links as edges. Given this network and initial probability of pages, the “PageRank” Algorithm Computes the Probability of Pages in the Graph through repeated iterations with a dampening factor. Let Page A has T1,T2,….., and Tm be the m pages referring to Page A (citations).

PR(A) = (1-d) + d (PR(T1)/O(T1) + ... + PR(Tm)/O(Tm))

PR(P) : Page Rank of Page P

O(P): Number of Outgoing links from Page P

Yet another method for refining web searches is Web Page indexing. It is done to collect and parse data from the stored web pages to facilitate fast and accurate “Information Retrieval”. Apache Solr is java based framework (Apache Lucene) typically used for web indexing. Term Frequency – TF(W) Number Of Times a Word W appears in a document; Inverse Document Frequency – IDF(W) Inverse of Number of Documents (Stored Pages) containing the Word W. Apache Solr, uses these for indexing.

**Objective:**

The relevancy of the retuned pages with respect to the queried words is good measure to evaluate search engines. With this set ahead, the goal is to compare and contrast different search engines such as Google, Bing, Apache Solr With Lucene, and Apache Solr with External Page Ranking (computed using NetworkX Library for Python).

**Other Goals:**

* To explore Solr
* Handle Solr indexing with external Page Ranking
* Compute Page Ranking using Directed Graphs and NetworkX library
* Create an interactive User-Interface for quering using Solr, and External PageRanking.

**SETUP and INSTALLATIONS**

**Setting up of Ubuntu 14.04 OS**

* I used VMWARE to create a virtual machine(vm) running 64bit ubuntu.
* Ubuntu iso was downloaded and installed into VMWARE.
  + 2GB RAM, 20GB Hard disk

**Setting up of Apache Tomcat and PHP in Ubuntu**

* I installed XAMPP V1.7 for my UBUNTU (vm) - a bundle for PHP, My SQL, Tomcat Server, and etc.
* Tested a Sample php. It was working well.

**Setting Up of Solr In Ubuntu**

* Downloaded Apache Solr 5.3.x from apache website
* Extracted and Compiled the sources as guided in the HW3 Description, to get the Solr Running
* Created a sample core “example core” and indexed the sample docs in the examples folder (under src directory).
* It worked well
* Deleted the sample core

**Installing NetworkX Package (Library) for Python 2.7 and above**

* Downloaded the Network Library by pulling it from the Github – <https://github.com/networkx/networkx/>
* Ran the ez\_setup.py and it made the job easier.
* Add those libraries to the $PATH Variable (in bashrc)

**WORK PRINCIPLE**

**Computing Page Rank**

**Tools** Used: Python; CSV and Text Files

Given:

Crawled Pages (Recently Crawled for HW3) – These are the pages that gets stored and indexed; pagerankdata.csv – Containing the Edges of the Network of Pages required to compute Page Rank, i.e., this comma separated version file contains a row for each outgoing link from the crawled pages; **docid.csv – this csv is used a map of Crawled Pages: URL of the Crawled Paged as the key and the id (name) of the Crawlded Page as the value. This csv is required for directing the link in the result table of UI to the original website from which the page was downloaded.**

Using this data and network library, I wrote a GraphGen.py python script that computes the page rank of pages in the graph and outputs two files (attached with the report)

* external\_pageRank.txt – Contains the id of the Crawled Pages along with the computed Page Rank
* external\_pageRank\_Complete.txt – contains the page ranks for all the nodes in the Directed Graph created from the pagerankdata.csv file. The URL of the link is used as the id of the page that is not crawled/indexed.

**Initial Values of Parameters of Page Rank Algorithms:**

G – Directed Graph computed from the pagerankdata.csv (had approximately 17631 nodes)

D = 0.85 – the dampening factor is not set too low (to avoid repeated over shoot) and also it is not high enough to limit the iterations.

N = 100 – the number of iterations is set to 100, though there were only a hundred thousands of edges (links) in the graph, and by rule of convergence O(log 200,000 ) << 100 is sufficient for the model to work.

**Indexing and Querying With Solr (TIKA – LUCENE/ External Page Rank)**

**Tools Used**: Solr; Page Rank File, Crawled Data, Solr XML Configuration Files

<all the commands and file destinations are with respect to the solr source directory>

Solr server that runs on jetty is started on a solo environment using the command – “bin/solr start”. A new core TikaCore3 is created using the command “bin/solr –c create TikaCore3”. The crawled Pages is stored under the folder “example/PagesDownloaded/”. Around 4577 pages were stored (A new crawl is made not on par with the crawl results of HW2).

The “managed-schema” configuration file under the core’s source folder is altered to combine all the results into one multivalued, stored, and indexed field “\_text\_”. This field is then set as the default field in SolrConfig.xml, just to make querying easy.

**I was lucky enough for not having to deal with de-duplication at the protocol levels. I checked the URLS of the pages crawled in the docid.csv and found that all urls are of “http” protocol. However, I configured the solrconfig.xml to handle exact de-duplication by inserted elements requesting UpdateHandler for Deduplication using “TextProfileSignature of Signature Class” that calculates a fuzzy hash of textual fields.**

The core is reloaded and indexing is done using the command “bin/post –c TikaCore3 examples/PagesDownloaded/”. It took some time for the pages to get indexed. Once the Indexing is done (checked through the log in the UI), sample queries were run to the indexing. It worked well.

The next step is to integrate pageRanking into Solr, this was done by placing the external\_pageRank.txt file in the core’s data folder. Further, the managed-schema.xml is modified to include the external field – pageRank of class “solr.ExternalFileField**”. The default value is set to 0.000035 (the minimum score minus 0.000000x from the computed pageRank). Since the pageRanking is limited to 4577 of 17631 pages, just to increase the relevancy level of the search results, I changed the default operator to “AND”.** Also the solrconfig.xml is configured by adding listeners on newSearch or FirstSearch. The TikaCore3 is reloaded and the results are sorted on pageRank desc. It worked well.

**Creating An Interactive UI for quering using Solr.**

**Tools Used**: JavaScript, Jquery, Ajax Calls, PHP, HTML, CSS

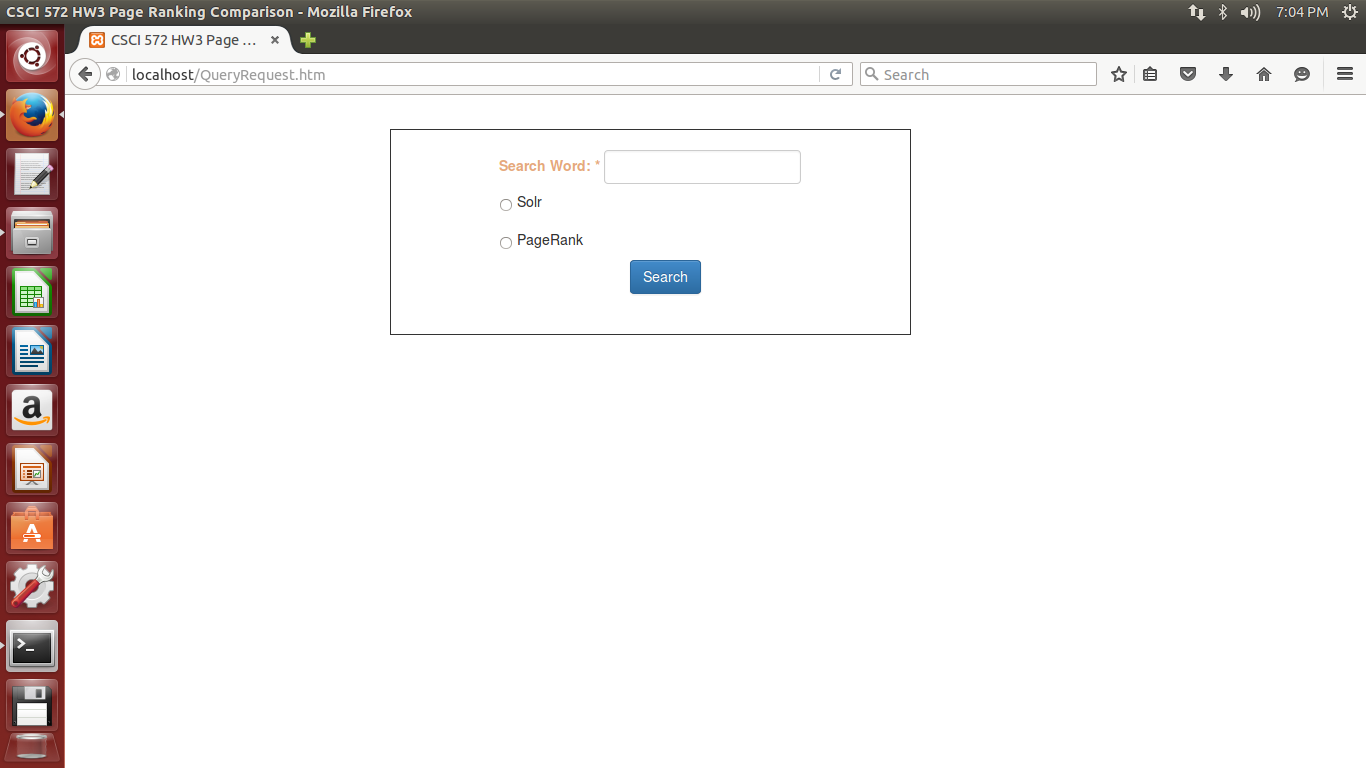
Created two separate files QueryRequest.htm and QueryProcessor.php (attached)

**QueryRequest.htm is an interactive UI for querying using Solr. It has the following features.**

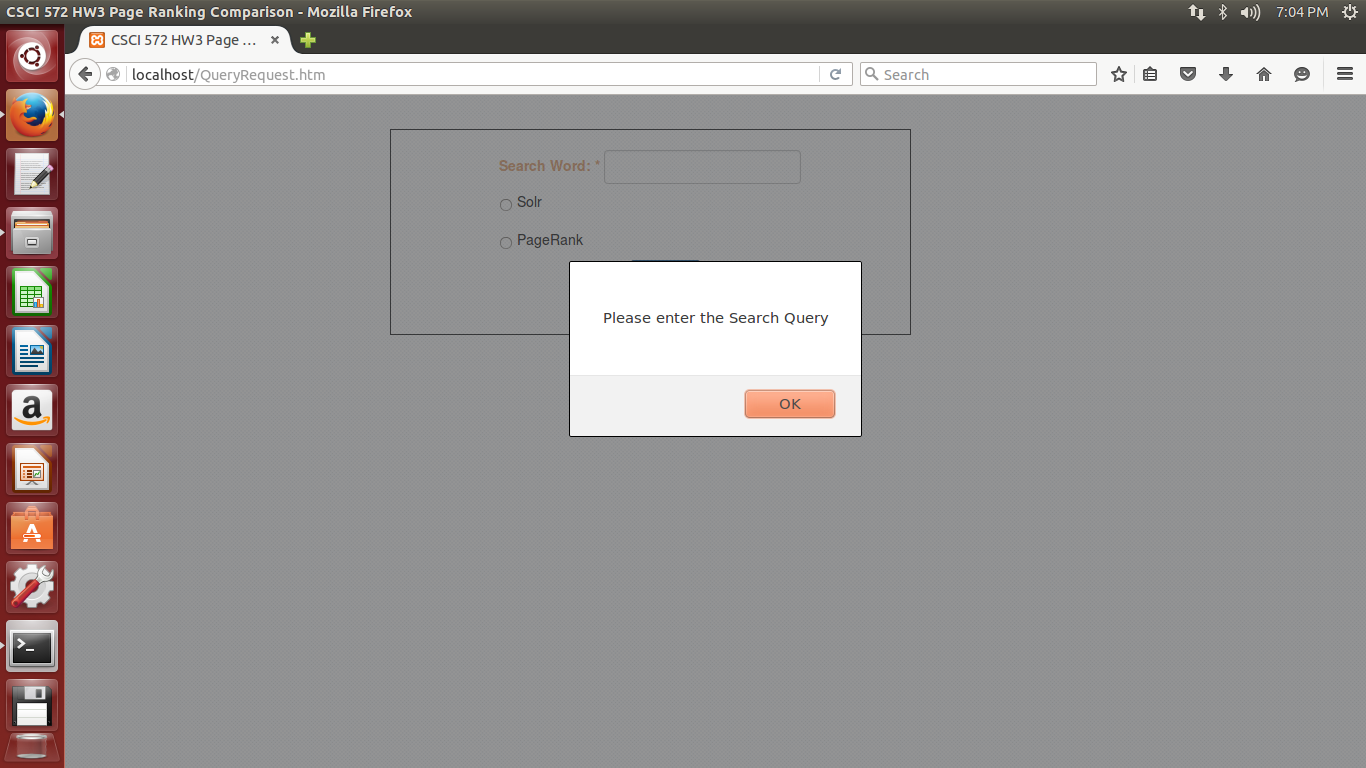
* Text box for querying
* Radio Buttons to toggle between Solr and External Page Rank
* Message Warning for Failure/Null Data Input.
* Upon Success a Table is displayed showing first 10 of the total results.
* Does not retain previous queries words/options upon refreshing
* Links are opened in a new tab

This takes the input entered through the text field and radio buttons, validates the input and makes an asynchronous ajax call to the QueryProcessor.php. The QueryProcessor.php gets the request processes it and queries the server based upon the radio button toggles. The result is returned as JSON Object. The QueryRequest.htm upon receiving the JSON Object, parses the objects and displays output in the specified format as seen from the screen shots.

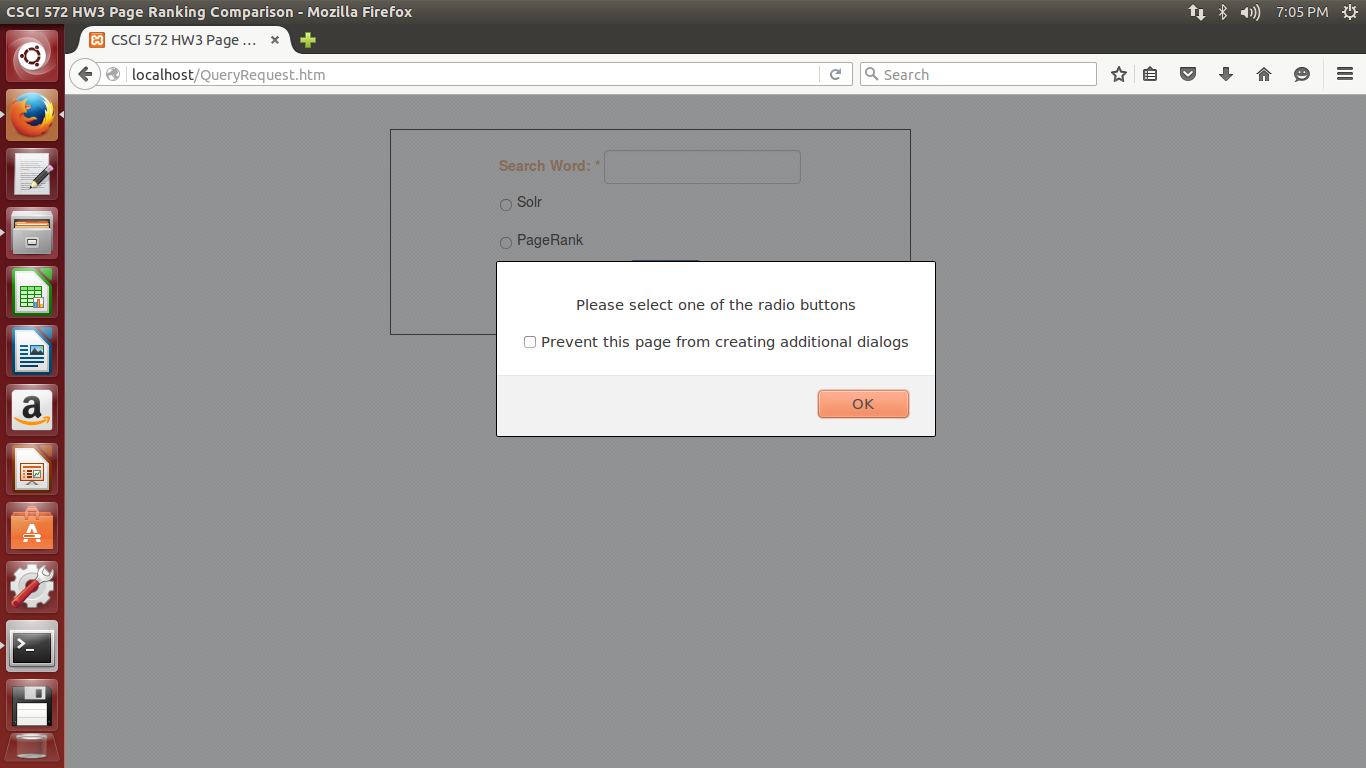
**Sreen shot of the Empty UI**



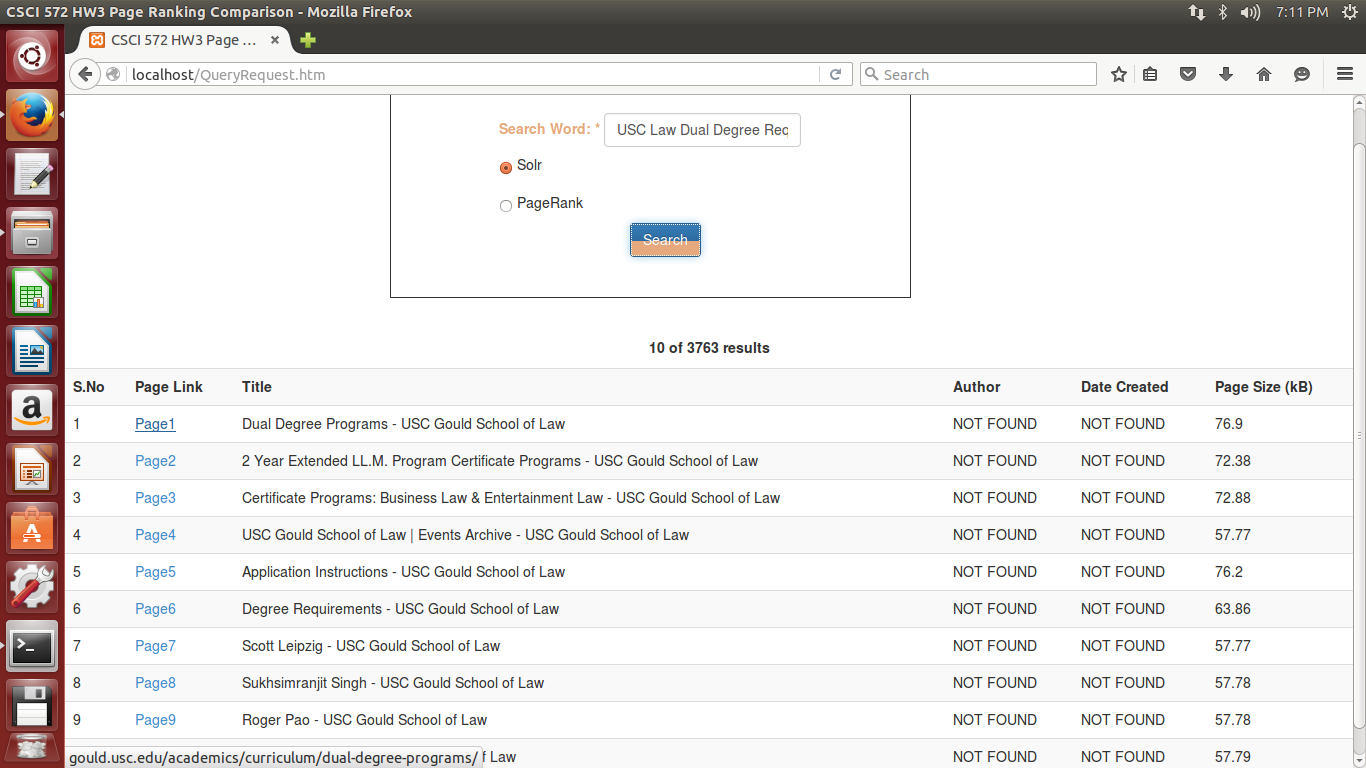
**Screen shot with Empty Search Field**



**Screen Shot with Unselected Radio Buttons**



**Screen Shot with Results for Solr**

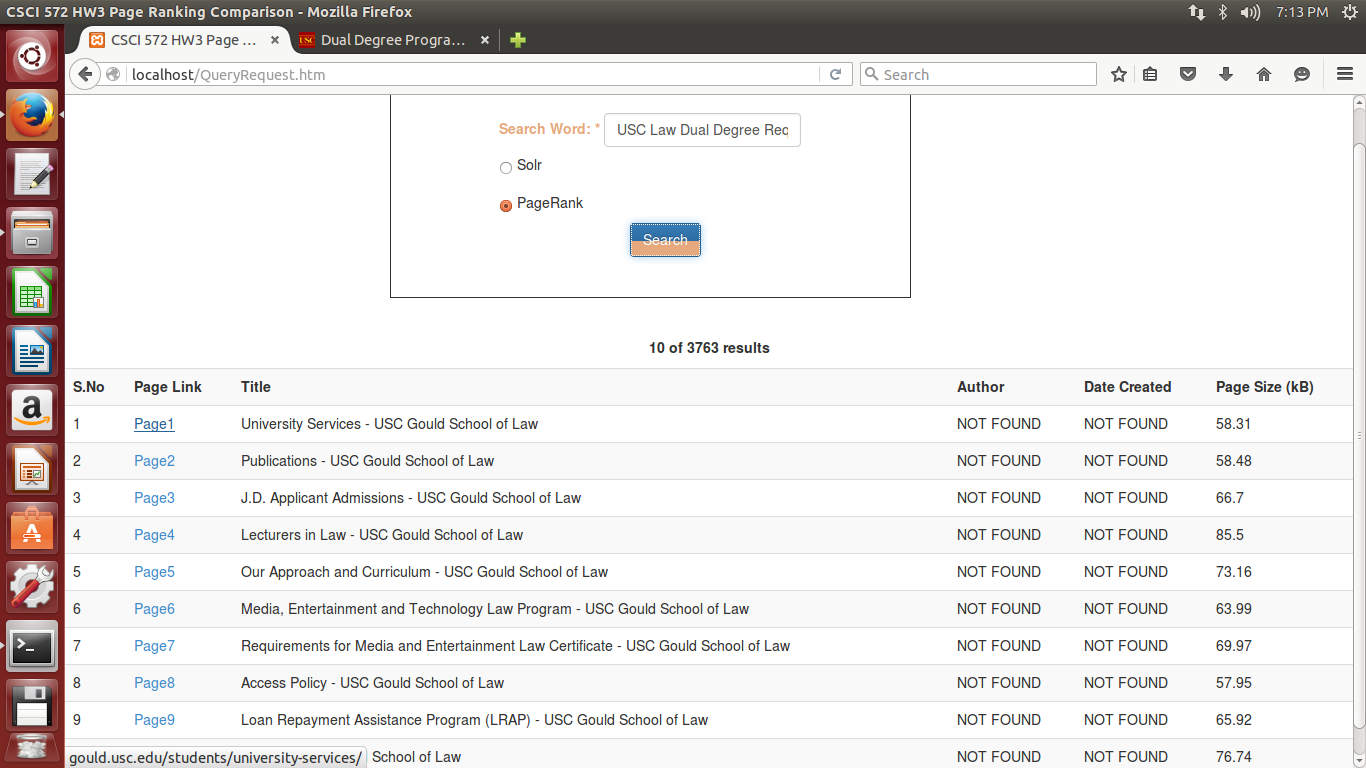


Verify that URLs in the snaps above and below for Solr

**Screen Shot for Clicking Page1 from the results of Solr**



**Screen Shot for Page Rank Results**



Verify that URLs in the snaps above and below for Page Rank

**Screen Shot for Clicking Page1 from results of Page Rank**



**Analysis of Results and Relevancy Graph Plotting**

The results of the chosen queries were compared and contrasted against GOOGLE, Bing, Apache solr with Tika And Lucene, and With External Page Ranking. A fresh querying is made to make this fair and square. “MyStatistics.xls” (attached with the report) is the excel sheet containing the graphs and formula used for calculating the relevancy for both navigational and informational Queries as follows (also attached with the report as “MyQueries.txt”).

The queries are enclosed within quotation marks

Navigational Qeuries

1.

a. "Scott Altman USC Gould School of Law"

b. "Pauline Aranas USC Gould School of Law"

c. "Jody David Armour USC Gould School of Law"

2.

a. "Center for Law and Philosophy, USC Gould School of Law"

b. "Center for Law, History, and Culture, USC Gould School of Law"

c. "Pacific Center for Health Policy and Ethics, USC Gould School of Law"

3. "Gould School of Law USC map and directions"

4. "Gould School of Law USC Founder"

5. "USC Gould School of Law Alumni"

Informational Queries

1. "USC Law Undergraduate Degree Requirements"

2. "USC Law Masters Degree Requirements"

3. "USC Law Dual Degree Requirements"

The two graphs for Navigational and Informational Queries can be found below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Navigational | | | |
|  | Google | Bing | Solr | PageRank |
| Relevant | 1 | 0.888889 | 0.555556 | 0.11111111 |
| Irrelevant | 0 | 0.111111 | 0.444444 | 0.88888889 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Informational | | | |
|  | Google | Bing | Solr | PageRank |
| Relevant | 0.633333 | 0.366667 | 0.133333 | 0.13333333 |
| Irrelevant | 0.366667 | 0.633333 | 0.866667 | 0.86666667 |

From the Graphs and table seen above, it is clearly evident that the Google has a better relevancy percentage on both grounds, i.e., for Navigational as well as the Informational Queries. Also Google and Bin Together has a far better results when compared to Solr and Solr with External page Ranking.

As already said, there are a lot of search algorithms influencing the relevancy of returned pages.

Google and Bing(Microsoft) being a Large Scale Industries Storing Humungous Data and Processing them for better results seems achievable when compared to limited resources on the Solr I used. My Explanation for the Relevancy Results and High Page Rank Values are as below

* **Limited Number of URLs Stored in the Search Spaces – affecting TF and IDF**

Google, Bing were able to index and compute Page Rank of Millions and Millions of URLs stored in their ornamented file systems designed for handling search oriented operations, whereas Solr that I used is restricted to 4577 Crawled Pages. Even the exernal\_pageRank.txt conatins the page ranks for the URLS that are crawled and downloaded in examples/PagesDownlaoded. The external\_pageRank\_Completed.txt contains the page ranks for all the pages in the Graph used for Computing the Page Rank. The number of pages in complete list is 17631 URLS where as the external\_pageRank that is used with SOlr has only 4577 URLs. The pages crawled and stored limits the values of Term Frequency (TF) and Inverse Document Frequency (IDF) that the Solar-cell (Tika) uses for indexing the pages.

Say PageA has the highest relevance to the query “Jody David Armour USC Gould School of Law” that refers to his profile. If PageA is not in the one of the 4577 pages. Then It won’t be indexed.

* **Limited Number of URLs Processed in Crawler**

In my example, the crawler fetched 4577 pages that contained around 17631 citation and cites including the crawled pages. This number is still too small to the daunting size of pages processed and tuned by Google, and Bing for optimizing search – around 300 Millions. If such is the ratio of URLs crawled and indexed in Solr to the URLs processed by Google, and Bing, then, the relevancy begin less than 0.33 for solr with or without external page ranking sounds ground.

* **Misleading PageRank Values**

Page Rank of a page A, PR(A) is computed using the formula mentioned above. The initial probability (default value) of the Page, Dampening factor, etc affects the Page Rank Value. Conceptually, it is the probability of number of citations to that page given the number of pages it cites to, predominantly dealing with number of incoming and outgoing edges of nodes in the directed graph (network). A naïve proportionality can be conjectured between the PR(A) and Number of Citations to A. Higher the Incoming edge to Page A(node A) the more is it’s Page Rank. The least a page is referred by, the least is its page Rank. That is why a few pages have higher Page Ranks that Other.It clearly entails that Circular paths in the graphs and also Self loops is a crucial factor impacting the Page Rank of A.

Given, this detail, Say Page X has the most relevance to Query Xq, but since the number of pages to be indexed is limited to 4577 (of 17631). The Page Rank of X is now confined to 4577 of 17631. Page X may have a poor score within this limit and thus making it not to appear in the top 10 results. The same Page X would have a higher Page Rank when computed by Google as they have millions and millions of URLS and in that graph Page X can have make citations thus making it more relevant. Computing page rank with 4577 is seeing the world from inside a dug-out well.

Just to minimize this misleading effect, I changed the defaule query processor operator to “AND” as can be seen in managed-schema.xml

* **Limited Resources:**

Google, Bing has huge storage and processing capacity against locally installed solr on Ubuntu VM Ware.