CS572 Informational Retrieval and Web Search Engines

Assignment 3 - Indexing the Web Using Solr

Steps Followed to Complete the Assignment:

I have downloaded only html, htm, pdf, doc, docx files only as a part of second assignment. The total downloaded files are around 2175.

I have also created the pagerankdata.csv file in HW2 which has successfully downloaded files in column1 and the subsequent columns contain the outgoing links contained within each downloaded page.

Step 1: Installing Ubuntu 15.10

I installed VMware Player from
https://my.vmware.com/en/web/vmware/free#desktop_end_user_computing/vmware_workstation_player/12_0

Downloaded the 64 bit version of Ubuntu Linux 15.10 from http://releases.ubuntu.com/14.04.

After downloading it, accessed the image through VMware player and installed Ubuntu in it.

Step 2: Downloaded solr-5.3.1.zip folder from http://lucene.apache.org/solr/quickstart.html.

Unzip the folder and type **bin/solr start -e cloud –noprompt** in terminal. This will launch the solr server on port 8983. Go to http://localhost:8983/solr and you should see the solr UI.

Create the core **bin/solr create –c myexample100** and modify the managed-schema.xml present in the **conf** directory by changing "text" to "_text_". Next uncomment the commented section in schema.xml.

After that index the downloaded files from the second assignment, using **bin/post –c myexample100 /home/varunmittal/Desktop/tmp1/**

Now we can check the Solr UI by opening the url http://localhost:8983/solr/. The number of files indexed is shown when the core is selected. The default querying option should be enabled by defining the default field in the requestHandler in the solrconfig.xml. It can be verified by running blank queries after reloading the core.

Step 3: Installed LAMP which will provide apache in the local system to host the localhost URL. For creating the localhost url, clone the solr-php-client to the home directory in my system i.e (/home/varunmittal) from the github using the command **git clone https://github.com/PTCInc/solr-php-client.git**. Next create the php file for showing the UI that requests the query, communicates with solr core and displays the first 10 results. The php file for the UI query is created in /var/www/html folder which is the root folder of Apache in which I have given the connection for connecting to the solr-php-client repository cloned above by giving the statement **require_once('/home/varunmittal/solr-php-client/Apache/Solr/Service.php');** and the connection to the core created in which we have stored and indexed the documents in Solr (my core name is myexample100)

Step 4: Write a python script that will build a netwrokx graph using the pagerankdata.csv file that was created during the second assignment. The graph contains a node and is connected to all its outgoing urls, the pagerank is calculated based on the number of incoming links. The external_pageRankFile.txt (which contains the indexed documents) and the external_pageRankFileComplete.txt (which contains all the pagerank score for all the URLs in the pagerankdata.csv file) is created by the netwrokx graph which was built using the python script using the PyCharm IDE.

After computing the page rank scores, place the text file in the data directory of the core. After that modify the managed schema to consider the extetnal_pagerankfile.txt which is handy for cases where you want to update a particular field in many documents more often than you want to update the rest of the documents

Next add the listeners, which will load the external file to solr while querying in the solrconfig.xml. Reload the core and run the query by mentioning "pageRankFile desc" in the sort field.

Explanation of initial configuration of PageRank:

PageRank computes a ranking of the nodes in the graph G based on the structure of the incoming links. It was originally designed as an algorithm to rank web pages.

PageRank is calculated by the calling the following function on the networkx graph.

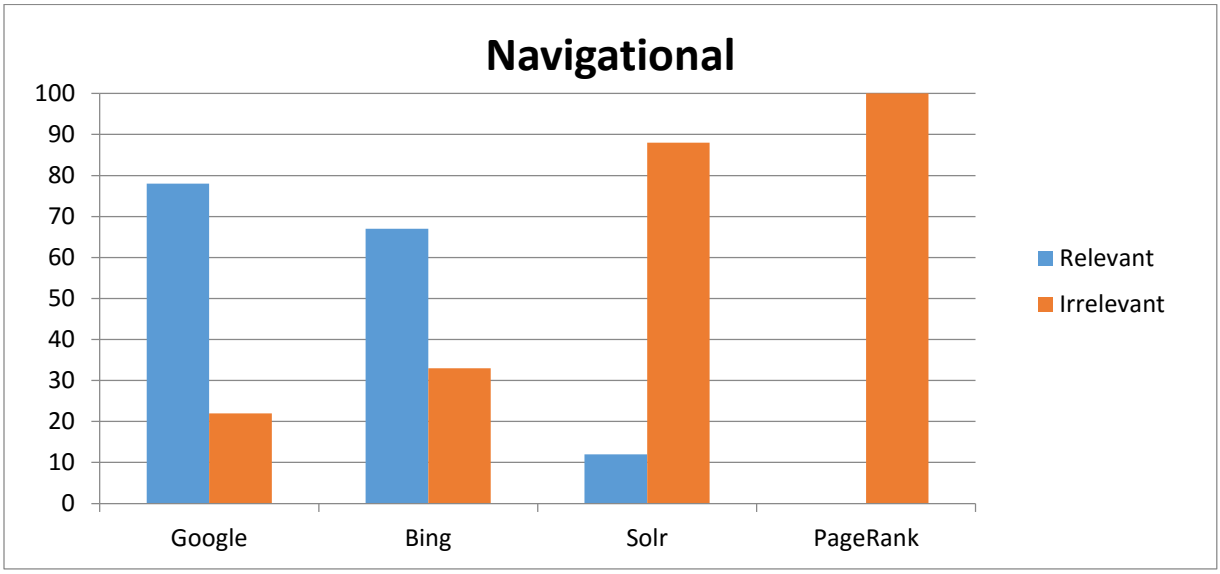
pagerank(G,alpha=0.85, personalization=None, max_iter=100, tol=1e-06,nstart=None, weight='weight', dangling=None)

Parameters:

- **G** (*graph*) – A NetworkX graph. Undirected graphs will be converted to a directed graph with two directed edges for each undirected edge.
- **alpha** (*float, optional*) – Damping parameter for PageRank, default=0.85.
- **personalization** (*dict, optional*) – The "personalization vector" consisting of a dictionary with a key for every graph node and nonzero personalization value for each node. By default, a uniform distribution is used.
- **max_iter** (*integer, optional*) – Maximum number of iterations in power method eigenvalue solver.
- **tol** (*float, optional*) – Error tolerance used to check convergence in power method solver.
- **nstart** (*dictionary, optional*) – Starting value of PageRank iteration for each node.
- **weight** (*key, optional*) – Edge data key to use as weight. If None weights are set to
- **dangling** (*dict, optional*) – The outedges to be assigned to any "dangling" nodes, i.e., nodes without any outedges. The dict key is the node the outedge points to and the dict value is the weight of that outedge. By default, dangling nodes are given outedges according to the personalization vector (uniform if not specified). This must be selected to result in an irreducible transition matrix (see notes under google_matrix). It may be common to have the dangling dict to be the same as the personalization dict.

The above function returns a PageRank which is a dictionary of nodes with PageRank as value. The function was used with all default values (pagerank(x, alpha=0.85)).

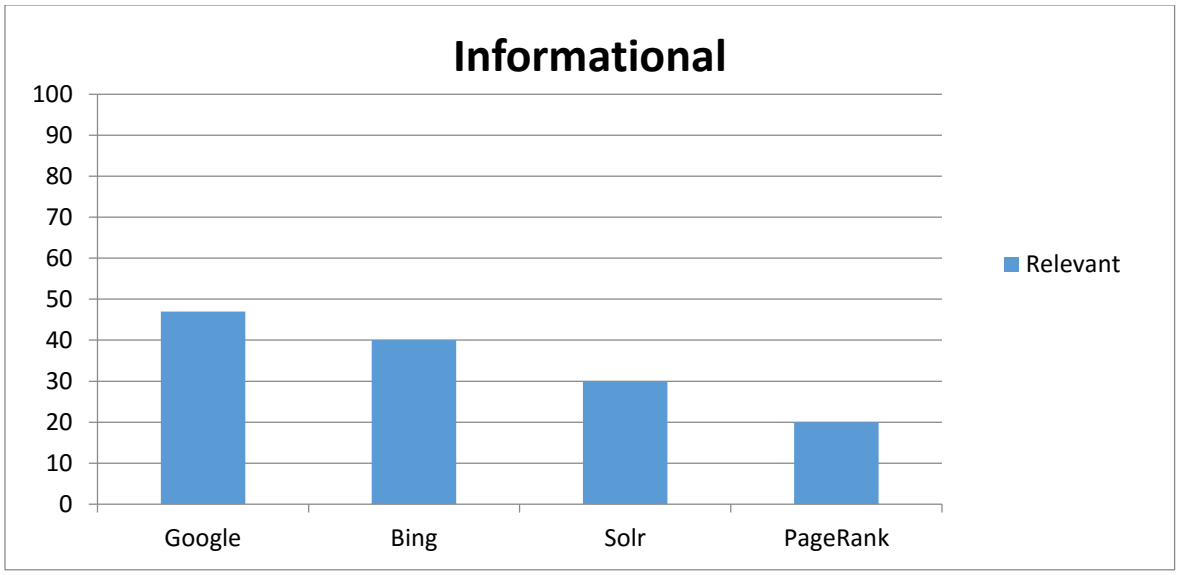
Analysis of Relevancy Graphs



When considering the first 10 results for a given query Google is performing better than Bing which is performing better than Solr which is performing better than PageRank as Google is giving a better percentage of relevant results as compared to Bing results as compared to Solr results as compared to PageRank results for navigational queries.

When considering the relevance for the navigational queries, there is no difference between the search engines

a) In case of navigational queries Google gives 78% relevant results as compared to Bing which gives only 67% relevant results
b) In case of navigational queries Bing gives 47% relevant results as compared to Solr which gives only 12% relevant results
c) In case of navigational queries Solr gives 12% relevant results as compared to PageRank which gives 0% relevant results



When considering the first 10 results for a given query Google is performing better than Bing which is performing better than Solr which is performing better than PageRank as Google is giving a better percentage of relevant results as compared to Bing results as compared to Solr results as compared to PageRank results for informational queries.

When considering the relevance for the navigational queries, there is no difference between the search engines

a) In case of informational queries Google gives 47% relevant results as compared to Bing which gives only 40% relevant results
b) In case of informational queries Bing gives 40% relevant results as compared to Solr which gives only 30% relevant results
c) In case of informational queries Solr gives 30% relevant results as compared to PageRank which gives 20% relevant results

According to my analysis Google is giving the best result of them all as compared to other search engines as it because tough queries set Google apart. which is more likely to yield relevant results in Google Search than Bing. That's not to say that Bing search engine can't handle difficult queries-it can-but it will take more time finding the right result in Bing than Google and hence yield less relevant results.

Bing is performing better than Solr as Bing is a public search engine with access to all the public documents whereas in Solr we have our own installation for searching the documents which is limited to the local machine space and as we have set a fetch limit of 5000 documents in the crawler in HW2 we are only able to download very few files as compared to Bing and google which have access to millions of documents online.

Solr is performing better than PageRank because as we know Solr uses Lucene classes to build an inverted index in which the index stores statistics about terms in order to make term-based search more efficient. This is because it can list, for a term, the documents that contain it. This is the inverse of the natural relationship, in which documents list terms.

We are just using PageRank as an external file by using the ExternalFileField field type in Solr makes it possible to specify the values for a field in a file outside the Solr index. For such a field, the file contains

mappings from a key field to the field value. Another way to think of this is that, instead of specifying the field in documents as they are indexed, Solr finds values for this field in the external file.

As we know external fields are not searchable. They can be used only for function queries or display. The ExternalFileField type is handy for cases where you want to update a particular field in many documents more often than you want to update the rest of the documents.

For example, suppose you have implemented a document rank based on the number of views. We might want to update the rank of all the documents daily or hourly, while the rest of the contents of the documents might be updated much less frequently.

Without ExternalFileField, we would need to update each document just to change the rank. Using ExternalFileField is much more efficient because all document values for a particular field are stored in an external file that can be updated as frequently as you wish and we can use pageRankFile desc to sort the indexed documents according to the score present in the external file for each document.

Reason for why certain pages have higher PageRank values

Google Page Rank is a tool designed to rank the sites included in the index of Google. High PageRank is determined by the number, relevance and importance of sites that link to your web page. When one page links to another, Google considers it as a vote for a web page.

The aim of Google is to provide the most relevant results for every query. Search results are generated based on the data search index. The search index is constantly changing. Updating Google ranking algorithms may change the position of URLs in the search results or delete them. Google cannot promise that any page will always be present in their index or have a certain ranking.

We can also get less relevant results which have higher page ranks because of multiple reasons like:

Internal Anchor Text

The most common issue I have seen when coming across this problem is the case of internal anchor text optimization gone awry. Many sites will have the keyword they're targeting on the intended page linking to another URL (or several) on the site in a way that can mislead search engines.

External Link Bias

The next most common issue I observed is the case of external links preferring a different page than me, This often happens when an older page on your site has discussed a topic, but you've more recently produced an updated, more useful version. Unfortunately, links on the web tend to still reference the old URL. The anchor text of these links, the context they're in and the reference to the old page may make it tough for a new page to overcome the prior's rankings.

Link Authority & Importance Metrics

There are times when a page's raw link metrics - high PageRank, large numbers of links and linking root domains - will simply overpower other relevance signals and cause it to rank well despite barely targeting (and sometimes barely mentioning) a keyword phrase. In these situations, it's less about the sources of links, the anchor text or the relevance and more a case of powerful pages winning out through brute force. On Google, this happens less than it once did (at least in our experience), but can still occur in odd cases.

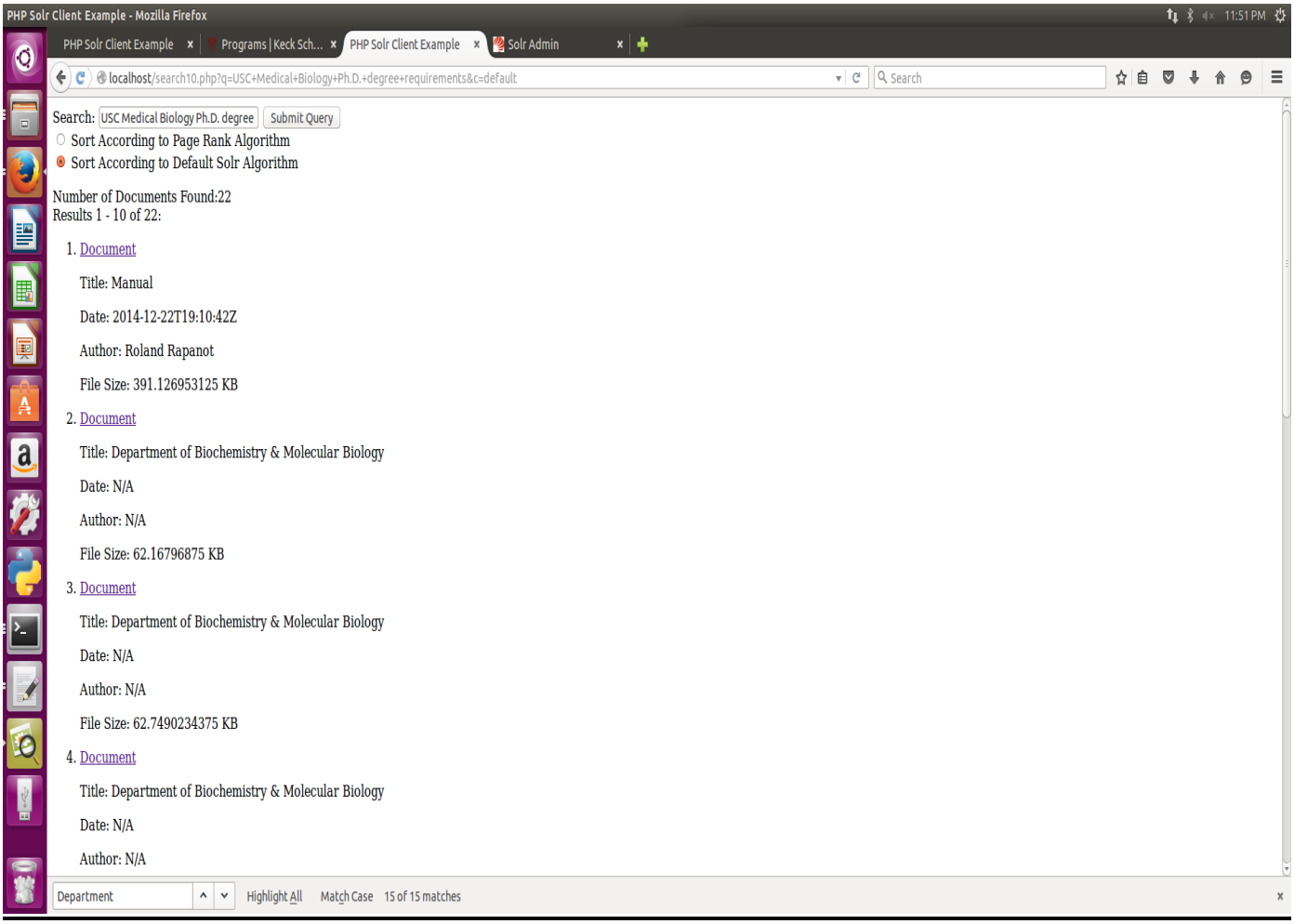
On-Page Optimization

In some cases, a webmaster/marketer may not realize that the on-page optimization of a URL for a particular keyword term/phrase is extremely similar to another. To differentiate and help ensure the right page ranks, it's often wise to de-emphasize the target keyword on the undesirable page and target it more effectively (without venturing into keyword stuffing or spam) on the desired page. This post on keyword targeting can likely be of assistance.

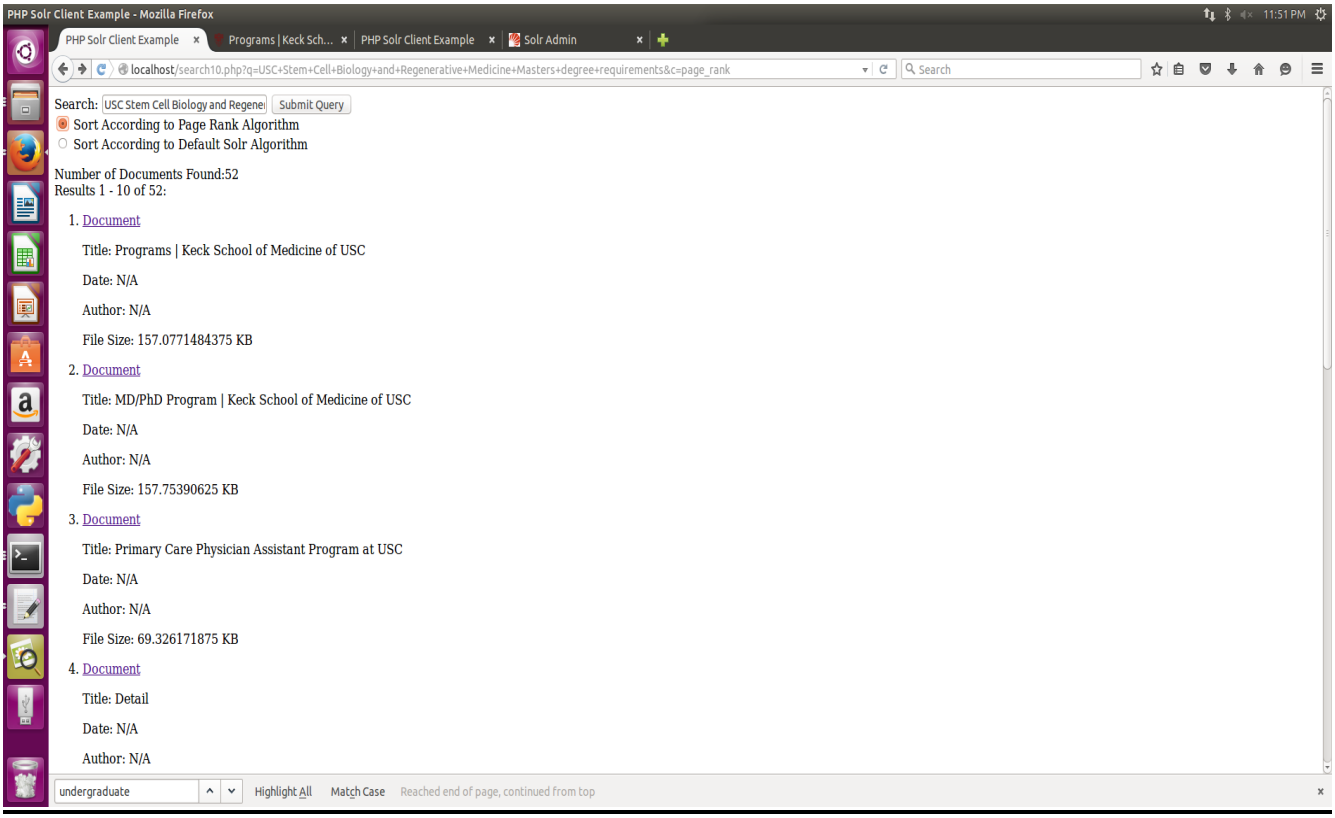
Topic Modeling / Content Relevance Issues

Essentially, we can think of the search engines doing a number of things to determine the degree of relevancy of a page to a keyword. Determining topic areas and identifying related terms/phrases and concepts is almost certainly among these. Seeing as this is likely the case, the engine may perceive that the page we're trying to rank isn't particularly "on-topic" for the target keyword while another page that appears less "targeted" from a purely SEO/keyphrase usage standpoint is more relevant.

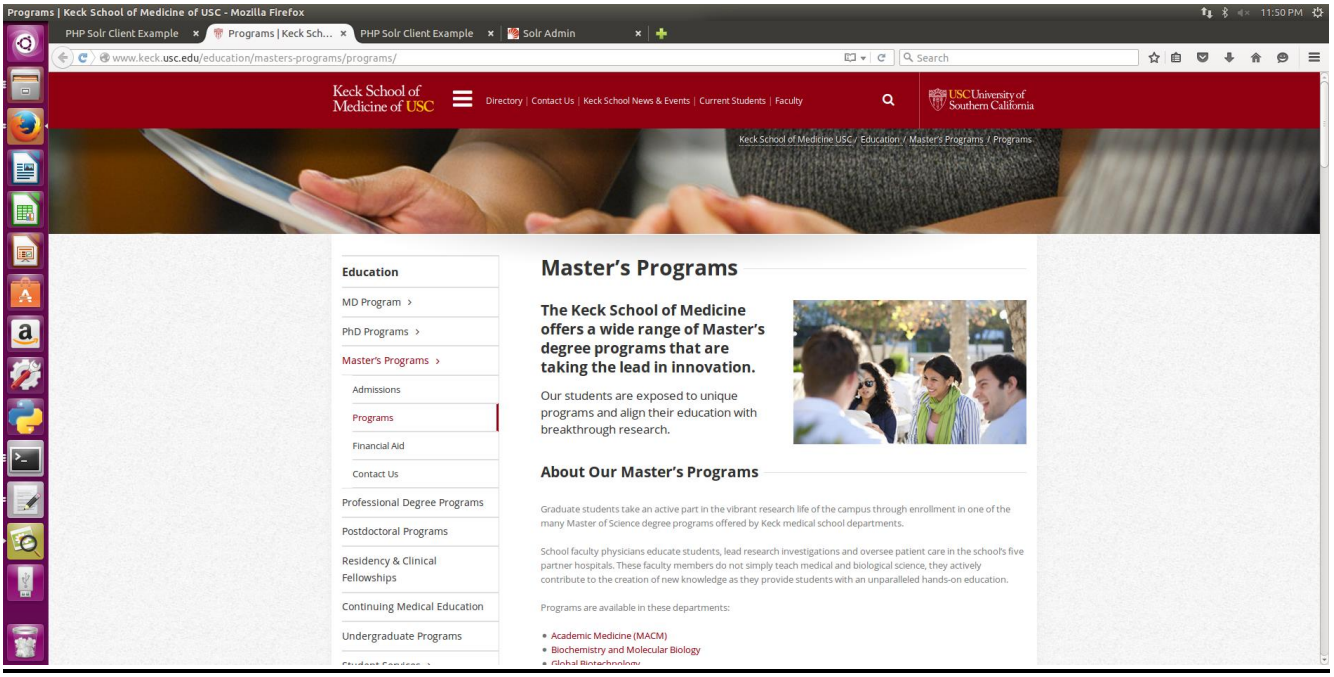
Screenshots:



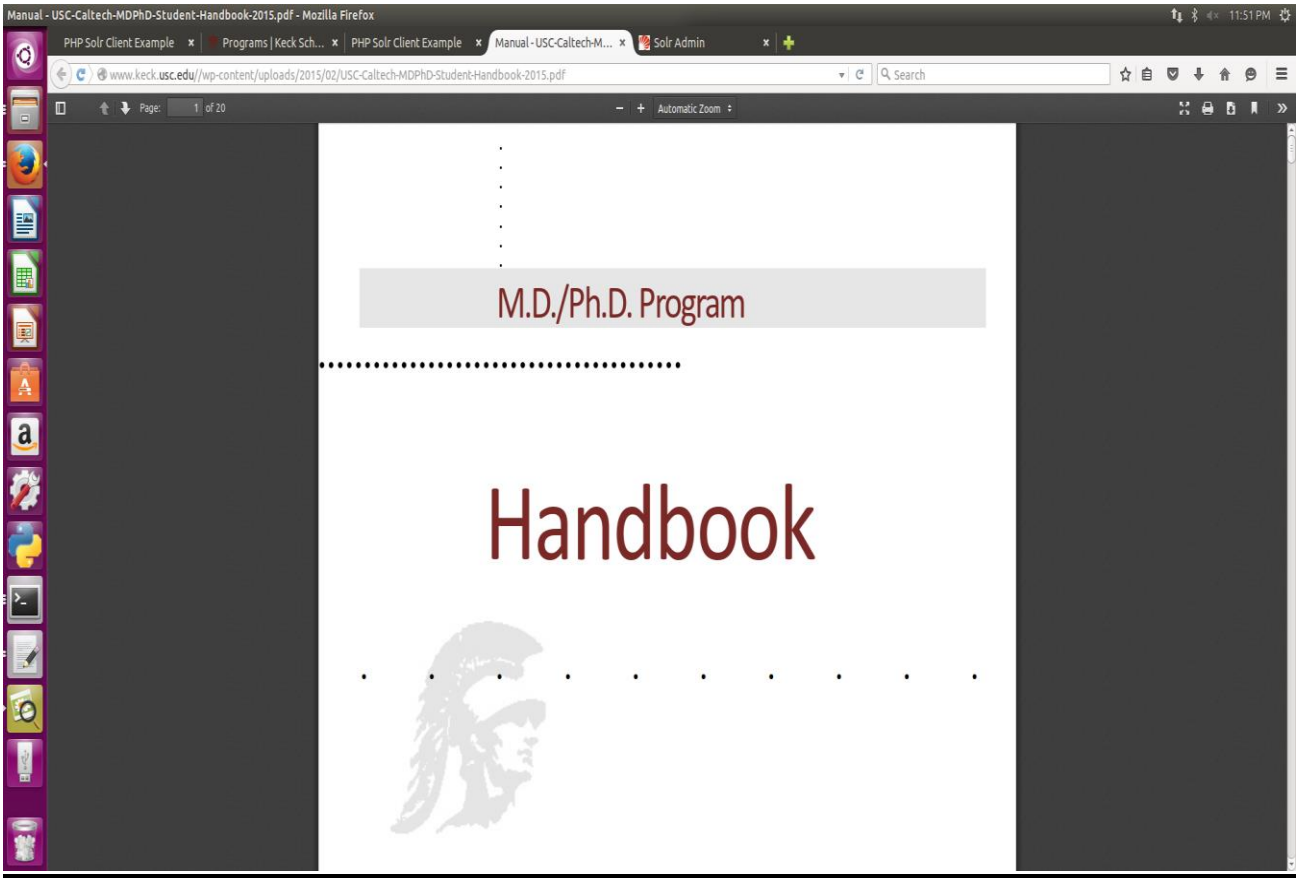
Query Entered in Solr



Query entered in PageRank



Document opened in Solr



Document opened in PageRank