The University of Texas at Dallas
CS 6322
Information Retrieval
Spring 2017

Class Project Proposal

Project TITLE: ***Search Engine for Animals***

Students:
Himanshu Gupta, hxg151930@utdallas.edu
Himanshi Aggarwal, hxa152730@utdallas.edu
Yashveer Singh, yxs156930@utdallas.edu
Karan Motani, kbm160230@utdallas.edu
Srabonti Chakraborty, sxc154030@utdallas.edu

Problem Statement: Search Engine for Movies

In order to implement our solution, we chose the following 5 sites –

The responsibilities for the project are as follows –

Crawling – Himanshu Gupta
Indexing and Relevance – Himanshi Aggarwal
UI – Yashveer Singh
Clustering – Karan Motani
Query Expansion – Srabonti Chakraborty

__**Application Architecture:**__

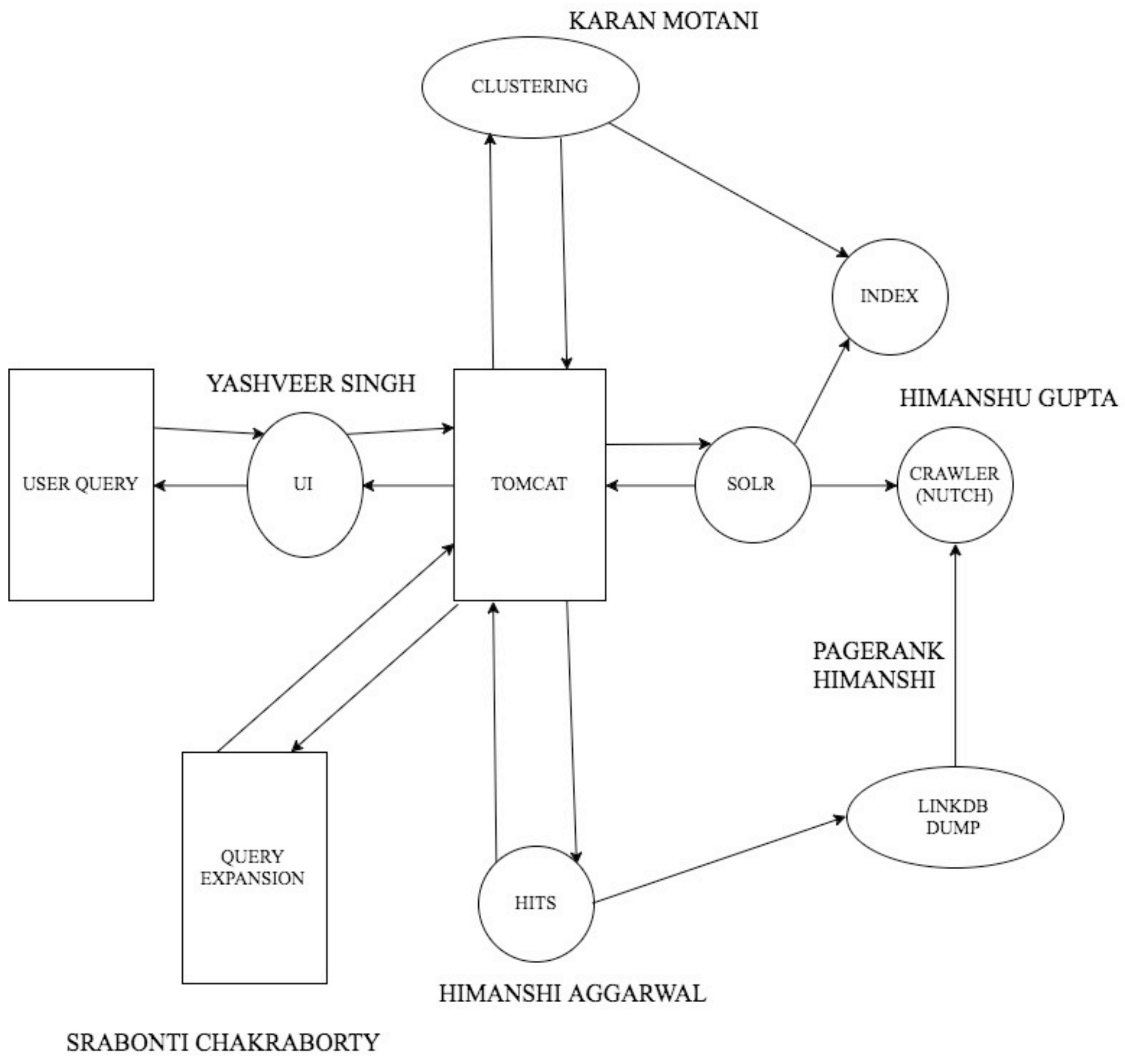


Figure 1: Architecture Diagram

1. **Crawling – Himanshu Gupta**

For crawling web pages associated with music, The Apache Nutch Framework was utilized for crawling as well as feeding the Fetched Content from crawling to the Solr Framework hosted on localhost for indexing the fetched web pages as well as creating web graphs for implementing Page Rank and the HITS algorithms.

The number of pages crawled were 1,032,856
The number of web pages crawled & fetched were 136,570

The Apache Nutch Framework was made to crawl Wikipedia.org, nationalgeographic.com, www.animalplanet.com, www.aboutanimals.com, animaldiversity.org, etc.

The Nutch Framework was fed **127** URLs as seed URL to start crawling. These URLs mainly constituted numerous animals, Zoos, National Geography, Animal Planet, Wildlife sanctuaries, Aquariums, Facts about animals, etc

The following are a sample list of URLs used:

http://kids.sandiegozoo.org/
http://kids.nationalgeographic.com/animals/
http://www.nwf.org/kids/ranger-rick/animals.aspx
http://pbskids.org/wildkratts/
http://www.animalfactguide.com/animal-facts/
http://www.softschools.com/facts/animals/
http://www.sciencekids.co.nz/sciencefacts/animals.html
http://www.enchantedlearning.com/Home.html
http://www.kidsplanet.org/factsheets/map.html
https://nationalzoo.si.edu/
http://www.nationalgeographic.com/animals/index/
https://www.worldwildlife.org/species/
https://www.fws.gov/
https://www.nwf.org/
http://www.wildanimalsanctuary.org/
http://www.sarveywildlife.org/
https://en.wikipedia.org/wiki/Category:Animals
https://a-z-animals.com/
http://www.kidzone.ws/animals/
http://eol.org/
http://www.animalfactsencyclopedia.com/
https://www.aboutanimals.com/

http://explorer.natureserve.org/
http://www.wildanimalsonline.com/
http://animaldiversity.org/
http://oceana.org/marine-life
http://kids.britannica.com/comptons/article-9272874/animal
http://www.refdesk.com/pets.html
https://www.factmonster.com/dk/encyclopedia/science/animals
http://www.worldanimalfoundation.org/index.html
https://answersingenesis.org/animals/
http://www.aquariumofpacific.org/
http://lib.colostate.edu/wildlife/
http://funfactsaboutanimalz.weebly.com/
http://www.aerialanimals.com/
http://habitatofanimal.blogspot.com/
https://en.wikipedia.org/wiki/Flying_and_gliding_animals
https://www.reference.com/pets-animals/
http://www.edurite.com/kbase/characteristics-of-aerial-animals
http://www.ducksters.com/animals.php
http://www.kidzone.ws/
http://www.earthlife.net/
https://animalcorner.co.uk/animals/
https://www.sciencea-z.com/main/UnitResource/unit/2/life-science/grades-k-2/animals
http://www.nationalgeographic.com/animals/
http://gws.ala.org/category/animals
http://www.animalfactguide.com/links/
http://www.kidsites.com/sites-edu/animals.htm
https://www.wildrepublic.com/en/animal-facts
www.animalplanet.com/pets/other-pets/small-pets/
www.animalplanet.com/pets
https://www.cor.net/index.aspx?page=716
http://www.animalplanet.com/pets/other-pets/unique-pets/
https://en.wikipedia.org/wiki/Pet
https://www.activityvillage.co.uk/pet-animals
https://www.britannica.com/animal/pet
https://www.britannica.com/animal

The Command for used for facilitating the Crawling procedure in Apache Nutch is:
bin/crawl urls/ Crawling/ http://localhost:8983/solr 3

- urls: The directory containing the "seeds.txt" file which contains the seed URLs for crawling.

- Crawling: The directory which stores the resultant directories generated by the Crawl procedure.
- http://localhost:8983/solr: the URL link for the Solr host to which the crawled web pages and their contents are fed for indexing by Solr.
- 3: The number of iterations for which the Crawl script is executed. In every iteration, the Crawl script keeps crawling and fetching contents from unvisited URLs in the CrawlDB.

Apache Nutch generates 3 folder during the crawling operation:

1. **CRAWLDB:** it maintains the information about URLs such as the fetch status, fetching schedule, metadata, etc.
2. **LINKDB**: For each URL, the LINKDB maintains the incoming and outgoing URLs for that URL which are further used to facilitate PAGE RANKING algorithm and the HITS algorithm.
3. **SEGMENTS:** contains multiple subdirectories within it. During Crawling, the crawl script creates multiple directory to store information for Crawl Fetching, Crawl Content, Crawl Parsing, Parsed Data and Parsed Text.

The Crawling Method can be described by the following methods:

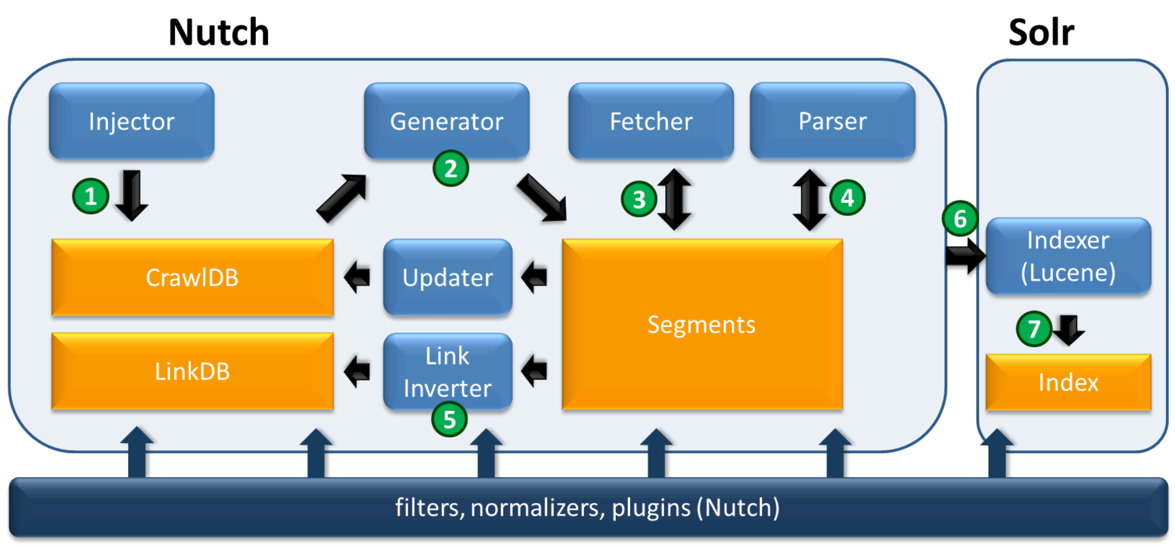
1. First Seed URLS are injected into the Crawl Database of Nutch. It maintains the list of URLs which are parsed by Nutch or are still in pending to be parsed.
2. Nutch next generates Segments for the injected Seed URLs. During the Crawl procedure, more than one segment may be generated. This helps Nutch determine which URLs have been crawled and parsed and maintains consistency with the REGEX filters.
3. Nutch begins Crawling and parsing the URL links in the Segments. It also stores information about the incoming and outgoing links in the URL.
4. When Nutch has finished crawling a set of URLs, these URLs are then added to the Crawl Database of Nutch. Also LINKDB is updated with the incoming and outgoing links of each URL.

To generate indexing, Page Rank and HITS algorithm, I collaborated with the student responsible for Indexing, PageRank and Hits. Next for implementing Page Rank and HITS algorithm, we shall require a Dump of the LINKDB inoder to utilize the incoming and outgoing links of every URL for implementing these algorithms. The command used for creating the Dump is:

```
bin/nutch readlinkdb Crawling/LinkDB -dump LinkDBDUMP
```

This generates a Dump containing the incoming and outgoing links for each URL.

Nutch – How It Works



1. The injector takes all the URLs of the nutch.txt file and adds them to the crawldb. As a central part of Nutch, the crawldb maintains information on all known URLs (fetch schedule, fetch status, metadata, …).
2. Based on the data of crawldb, the generator creates a fetchlist and places it in a newly created segment directory.
3. Next, the fetcher gets the content of the URLs on the fetchlist and writes it back to the segment directory. This step usually is the most time-consuming one.
4. Now the parser processes the content of each web page and for example omits all html tags. If the crawl functions as an update or an extension to an already existing one (e.g. depth of 3), the updater would add the new data to the crawldb as a next step.
5. Before indexing, all the links need to be inverted, which takes into account that not the number of outgoing links of a web page is of interest, but rather the number of inbound links. This is quite similar to how Google PageRank works and is important for the scoring function. The inverted links are saved in the linkdb.
6. and 7. Using data from all possible sources (crawldb, linkdb and segments), the indexer creates an index and saves it within the Solr directory. For indexing, the popular Lucene library is used. Now, the user can search for information regarding the crawled web pages via Solr.

In addition, filters, normalizers and plugins allow Nutch to be highly modular, flexible and very customizable throughout the whole process. This aspect is also pointed out in the picture above. I hope, this post helped you getting a basic understanding of the inner workings of Nutch.

2. **Indexing and Relevance – Himanshi Aggarwal**

2.1 Indexing

For indexing we used apache solr. After running the crawler, we get our output in three folders which are linkdb, crawldb and segments. Apache solr makes use of these 3 folders and generates the index of the documents

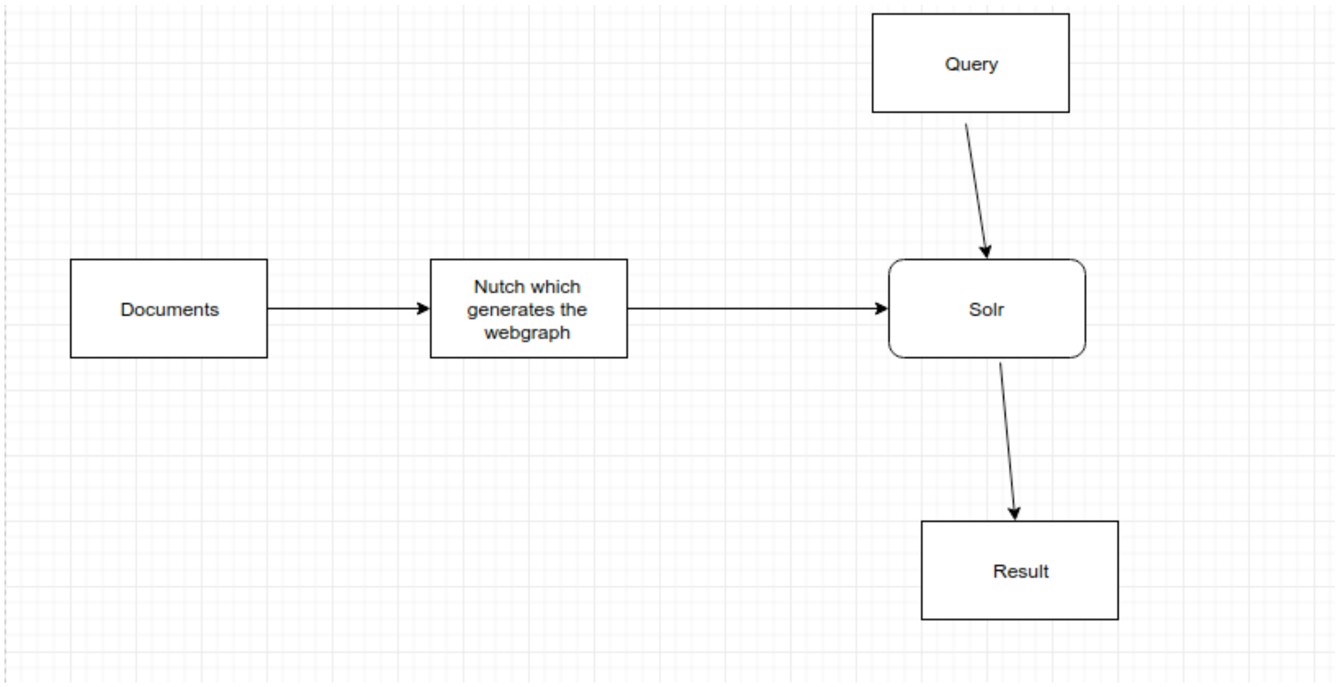
The commands used to feed the three folders (linkdb,crawldb and segments) is:

```
bin/nutch index -Dsolr.server.url=<url to the solr server>  <path to crawldb> -linkdb
<path to linkdb> -dir <path to the segmenst folder>
```

2.2 PageRank

To generate pagerank we used a webgraph which contains the number of inlinks, outlinks and the score of a given node using the pagerank algorithm. This webgraph is generated by nutch and is then fed into solr, which takes into account the TF-IDF score and the pagerank score in the webgraph to display the relevant results. (The damping factor here is 0.85)

The below figure shows the how recommendation takes place



The following steps were involved in recommending results:

a. Generate the webgraph using nutch. The comsmand used is : db bin/nutch org.apache.nutch.scoring.webgraph.LinkRank -webgraphdb crawl/<name of the webgraph folder>
b. Update the pagerank score of the crawldb, using the command:
bin/nutch org.apache.nutch.scoring.webgraph.ScoreUpdater -crawldb <path to crawl db> -webgraphdb <path to webgraphdb>

c. Index the documents using Apache solr using the command:
bin/nutch index -Dsolr.server.url=<url to the solr server> <path to crawldb> -linkdb <path to linkdb> -dir <path to the segmenst folder>

Total Number of links = 541240
Total Number of nodes = 121100
The largest number of ingoing links = 183259
The largest number of outgoing links = 26

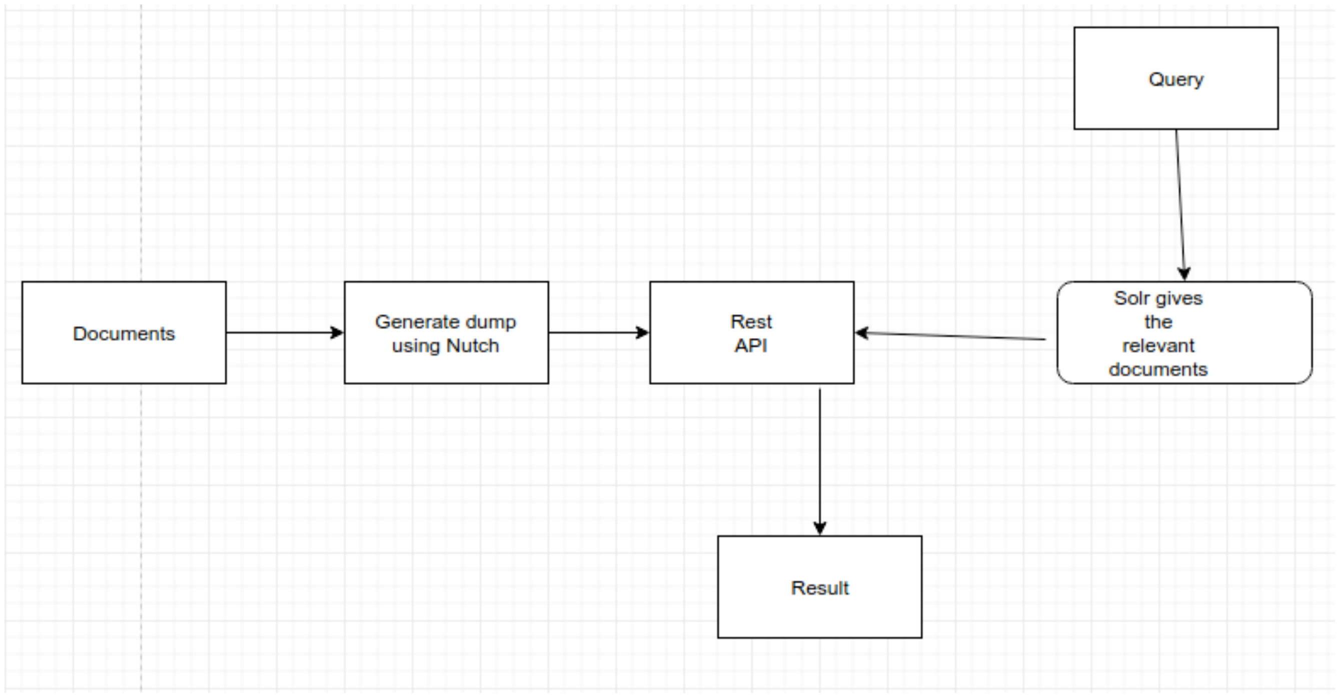
2.3 HITS Score

For the HITS score we made our own rest API and wrote our own code in python to generate the authority and the hub score in python.

In python we used the **networkx library** to create the graph and ran the hits algorithm on the graph to generate the hub and the authority scores. To generate the graph, we needed the inlinks and outlinks to a given document. To get these inlinks and outlinks we used nutch and generated a dump of the all the documents crawled. To do this we used the command:

```
bin/Nutch readlinkdb <linkdb> -dump <dump_directory>
```

The below image shows how hits score is generated and how it is used to show the results



The highest hubscore was assigned to:
https://www.wikidata.org/wiki/Q144 - (0.3180)

The highest authority score was assigned to:
https://www.wikipedia.org/wiki/Hunting_Dog - (0.2759)

Collaboration –

i. Clustering- Wrote the java code to generate retrieve the data from solr and generate a file which contains the (url, content).
ii. Crawling- I got the segment, crawldb and linkdb from the person who was doing crawler.
iii. UI- The pages generated from solr were already pagerranked and to obtain the hits score, I wrote a rest API in python which accept the urls and gives the authority and hub score for each of the urls.

3. **User Interface** – Yashveer Singh

Technologies Used: HTML5, JavaScript, jQuery, CSS, Bootstrap, Java Servlets

3.1 Design:

The system is designed as a multi-layer architecture. The following are the components of the architecture.

3.1.1 Presentation Layer (UI Layer):

The layout of the front end was designed using HTML5 and JavaScript (jQuery and Bootstrap). The UI has the following components embedded:

1. Logo of the search engine
2. Search text box with a search button
3. Frames for Google, bing and our results.

Here are the following tabs:

3.1 *Google* – Results from Google search API

3.2 *Bing* – Results from Microsoft Bing Search API

3.3 *Vector Space + Pagerank*:
Results from Solr server after indexing using vector space model and pageranking

3.4 *Vector Space + Pagerank + HITS*:
Results from HITS server after applying HITS over pagerank results from Solr

3.5 *Vector Space + Pagerank + Flat Clustering*:
Results from Clustering server after applying K-means clustering over Pagerank results

3.6 *Vector Space + Pagerank + Query Expansion*:
Results from Query Expansion server after expanding the query using relevance feedback

All search requests are sent to the server via AJAX requests and the responses are handled inside JavaScript.

3.1.2 Application Layer (Controller layer):

The search requests from the user are handled by the application layer which accepts GET and POST requests via AJAX calls from UI. There are six controller programs written in Java which performs the handling of AJAX calls.

ServletHandler (Controller 1):
Handles AJAX requests related to Google, Bing, Solr and HITS

ClusterServletHandler (Controller 2):
Handles AJAX requests related to K-means clustering

QueryExpansionServletHandler (Controller 3):
Handles AJAX request related to Query Expansion

3.1.3 Data Access Layer:

The requests handled via controllers are routed to appropriate servers and the response received are routed back to the Controller layer.

3.1.4 Business layer:

There are 6 servers running locally that contains the business logic for various relevance models.

Server 1: Solr server for handling indexing and Pagerank requests
Server 2: Python server for handling HITS/query expansion requests
Server 3: Python server for handling Flat Clustering requests

All requests from the Data Access layer are routed to these six servers through the APIs exposed by these servers. JSON is chosen as the data exchange format for ease of implementation and support by various programming languages.

3.2 Collaborations

3.2.1 Collaboration with student responsible for Indexing/Pagerank/HITS:

Indexing and pageranking are done using Solr module. The server running the Solr module exposes an API for accessing the indexed results. Using the information provided by the student about the Solr API and its requirements, the API was accessed from the DataAccess Layer via an exclusive call to the Solr server. This API exposes various control parameters such as number of pages to return and the format of the response expected. Owing to the scope of the project to study the effects of various relevance models are their comparison with Google and Bing, the number of results fetched was limited to 15 per server.

Result format (Common results format used through out the project): The JSON results from the Solr contain various information about the pages being retrieved. The following fields are selected for display in the UI:

1. URL: URL address of the page
2. Title: Title of the page
3. Content: the HTML content of the page (with HTML tags removed) – truncated to 200 characters for display in the UI as a short description about the webpage

The HITS algorithm is implemented in python and hosted in a separate server by this student. So the information about this python API was collected from this student to make the API call. Since the HITS algorithm is dependent on the indexing results from Solr, results from Solr API was sent to this API via POST and the response was collected back.

3.2.2 Collaboration with student responsible for Clustering:

The clustering algorithms are implemented and the cluster information was stored in separate files. These files are given to me by the student responsible for clustering and they are read to find the most relevant cluster for the user query by analyzing the top 100 results from Solr. The top 15 results were shown in the UI based on the chosen cluster. So the results being displayed would be ordered based on their relevance to each other.
The response format used is similar to the format used for Solr API.

3.2.3 Collaboration with student responsible for Query Expansion:

The query expansion algorithm is implemented and hosted in the same server as HITS. The input to this Query Expansion API is the initial query string made by the user and the initial Solr output. Final output from the API is the expanded query string. This expanded string is then used to retrieve a new set of results from the Solr API.

3.3 Comparison with Google and Bing:

In order to compare the results with Google and Bing results, Google web search API and Bing Search API were used to retrieve their results. Following observations were made during the evaluation:

a. For certain queries which contained Animal name or Zoo or Aquariums, our results from Pagerank/HITS/Clustering were closely in accordance with results from Google and Bing. We believe this is due to the selection of very good seed URLs to cover most of the albums and their artists. Also, from the type of pages that has been crawled, it is observed that the information about artists or their albums is static information and has good keywords. For example, popular pages about the animal 'dog' helped us to retrieve them and rank them higher.

b. For certain queries requesting animals or species from a particular region, our search engine did not retrieve relevant pages in comparison to Google and Bing. Upon analyzing the results from Google and Bing with our document collection, it is observed that the websites that are popular for such information like nationalgeographic.com denied access to our crawler. Hence those pages were missing from our index.

c. Using seed URLs from Google and Bing proved useful in retrieving the top pages during the search. Especially, Wikipedia pages were immensely useful in improving our search results and were retrieved by our search engine despite the limited number of inlinks and outlinks in comparison to the entire web used by Google and Bing. This proved the effect of Vector Space model combined with Pagerank.

3.4 Testing strategy:

Testing was done throughout the development of the project to provide immediate feedback to students implementing the relevance models. Approximately 200 queries were used to test the results of various models. About 50 queries about Animal names, Zoo names, Jungle and Aquariums were used in collaboration with the students building the relevance models and the rest of queries were generated based on various topics such 'animal name', 'particular species', 'zoos', 'aqariums', etc.

When a particular page from the top results of Google/bing does not come up with our relevance model, the page was first checked for existence and if present, the page rank score of the page was compared with the top scoring page from our relevance model. In few cases, the score was too low due to insufficient inlinks and outlinks and in many cases, the page in question was not present in our collection.

In HITS and Query expansion testing, certain times, the algorithms did not converge within the given number of iterations and were communicated to the student in charge and algorithm parameters were appropriately improved.

UI – Yashveer Singh, yxs156930

3.5 Results

The queries for demonstration were chosen from the testing we have done and gave good results. Each query chosen was targeting a different area such as Animal names, Zoo's, Aquarium, etc.

Query 1: *Zoo*

Top 5 results from our model (after Clustering) :

https://en.wikipedia.org/wiki/Zoo
https://detroitzoo.org/visit/today-at-the-zoo/
http://www.sfzoo.org/
https://detroitzoo.org/
https://detroitzoo.org/about/greenprint-sustainability/

Top 5 results from Google:

http://zoo.sandiegozoo.org/
http://www.oregonzoo.org/
https://nationalzoo.si.edu/
http://www.imdb.com/title/tt3250026/
http://www.houstonzoo.org/

Top 5 results from Bing:

www.cbs.com/shows/zoo
www.imdb.com/title/tt3250026
www.nczoo.org
www.philadelphiazoo.org
www.houstonzoo.org

Query 2: Jungle

Top 5 results from our model (After Hits):

https://answersingenesis.org/animal-behavior/jungle-doctors/
http://www.animalfactsencyclopedia.com/Jungle-animals.html
https://en.wikipedia.org/wiki/Jungle
https://answersingenesis.org/wild-brothers/
https://a-z-animals.com/quiz/polar-animals-quiz/

Top 5 results from Google:

https://en.wikipedia.org/wiki/Jungle
https://www.facebook.com/jungle4eva/
https://www.youtube.com/watch?v=GfxcnX7XWfg
https://soundcloud.com/jungle-8
http://www.imdb.com/title/tt3758172/

Top 5 results from Bing:

www.junglejunglejungle.com
https://en.wikipedia.org/wiki/Jungle
https://www.merriam-webster.com/dictionary/jungle
www.thefreedictionary.com/jungle
www.dictionary.com/browse/jungle

Query 3: Dog

Top 5 results from our model (after PageRanking):

https://www.wikidata.org/wiki/Q144
http://www.bestanimalsites.com/dog-houses.html
http://www.digitaldog.com/award.html
http://www.digitaldog.com/choosing.html
http://www.digitaldog.com/contactus.html

Top 5 results from Google:

https://en.wikipedia.org/wiki/Dog
http://www.vetstreet.com/dogs/
https://www.dog.com/
http://www.animalplanet.com/pets/dogs/
http://pets.webmd.com/dogs/

Top 5 results from Bing:

https://www.dog.com
https://simple.wikipedia.org/wiki/Dog
https://en.wikipedia.org/wiki/Dog
www.thefreedictionary.com/dog
www.nextdaypets.com/directory/breed

4. **Clustering – Karan Motani**

Input:
Input is a file with two columns [URL, Content]. The URL represents the id and the content contains the content of the webpage.

Output:
Output of this module is a file which contains the URL and the respective cluster it belongs to.

Collaboration:
With the student who Implemented Indexing:
We together developed a code to get the data from the solr was fetched and it was parsed such that to generate a file with just the URL and the content of the webpage, each represented in a single line.

With the student who Implemented the UI:
The clustering helps in finding similar pages based on the contents of the webpage. So we decided up on the using this property by first retrieving the query results from the Page Ranking. The top 100 of the Page Ranking results are checked with the clustered results by comparing how many of these webpages belonged to same clusters and we consider the cluster which contains the maximum webpages and display 15 of the webpages on to the UI.

Models Used:

I have used both Flat Clustering and Agglomerative Clustering (Crashes at end).

Flat Clustering:
Kmeans algorithm was used to implement the flat clustering.
Tool used: Python Sklearn library was used to generate the clusters.
Parameter Values: Number of cluster = 100, max_iterations = 100

Taking the time complexity and the accuracy we were obtaining we decided 100 clusters as optimal.

Agglomerative Clustering:
Two Agglomerative clustering have been implemented (Crashes at end).

Complete-link and Average methods
Tools used: Python SciPy library was used to generate the clusters.

Python Scipy library generates a file which contains the URL ids which are connected in the tree format.

Running agglomerative clustering requires O(n^2) memory complexity. With our total of 170000 webpages we had a memory constraint. Because of huge amount of pages, we could not generate clusters using Agglomerative Clustering algorithms as the algorithm was taking a long time and crashed at the end.

Problems Faced:

Time and Memory complexity of the algorithms in use is the major problem faced here. Agglomerative Clustering using centroid method was taking a long time (close to 6 hours for first 50000 webpages) and crashing at the end which could not generate the clusters before crashing and hence not shown in the UI.

Results:

Around 30-40 queries were used to test the clustering results to get the parameters right. A few test cases are displayed below.

1. **Query: dog**

Clustering Results:

https://www.bestanimalsites.com/dog-houses.html
https://www.localdogwalker.com/
https://www.localdogwalker.com/pet-sitting/
https://www.pettravel.com/
https://www.bestanimalsites.com/keeping-dog-safe-in-hot-weather.html
https://en.wikipedia.org/wiki/Bluey_(dog)

2. **Query: cat**

Clustering Results:

https://en.wikipedia.org/wiki/Cat-burning
https://en.wikipedia.org/wiki/Munchkin_cat
https://en.wikipedia.org/wiki/Persian_cat
https://en.wikipedia.org/wiki/Sand_cat
https://en.wikipedia.org/wiki/Sphynx_cat

3. **Query: Zoo**

Clustering Results:

https://en.wikipedia.org/wiki/Zoo
https://detroitzoo.org/visit/today-at-the-zoo/
http://www.sfzoo.org/
https://detroitzoo.org/
https://detroitzoo.org/about/greenprint-sustainability/

4. **Query: Gorilla**

Clustering Results:

https://a-z-animals.com/animals/western-lowland-gorilla/
https://a-z-animals.com/animals/western-gorilla/
https://a-z-animals.com/animals/cross-river-gorilla/
https://a-z-animals.com/animals/eastern-gorilla/
https://a-z-animals.com/animals/gorilla/

5. **Query: whale**

Clustering Results:

http://www.akwhalewatching.com/
http://www.iucnredlist.org/details/15421/0
https://en.wikipedia.org/wiki/Northern_right_whale
http://acsonline.org/fact-sheets/orca-killer-whale/
http://eol.org/data_objects/13230845

5. **Query Expansion – Srabonti Chakraborty**

Rocchio Algorithm:
Rocchio Algorithm reformulates a query in such a way that:

It gets closer to the neighborhood of the relevant document and gets away from the neighborhood of the non-relevant documents. Documents identified as relevant (to a given query) have similarities among themselves and non-relevant docs have term-weight vectors, which are dissimilar from the relevant documents.

Twenty queries have been selected to verify the Rocchio algorithm. Following was the approach used to arrive to a set of weights that fetches a more relevant modified query:

1. Get the query and the results from Solr for the results for the query
2. Fetch the the results from Google
3. Label each result from our search query as either relevant for non-relevant based upon the results of Google.
4. Increment beta, decrement gamma keeping alpha constant to get higher ratio.
5. Repeat step 4 for a fixed number of iterations or until weights stabilize
6. Output the modified query from the query vector

Those twenty queries are selected as follows:

1. 5 queries with one keyword as the query for data that was crawled
   a. tiger
   b. elephant
   c. dog
   d. dolphin
   e. squirrel

2. 5 queries with two keywords as the query where one is animal and another is animal related
   a. Pet dog
   b. River fish
   c. cuckoo nest
   d. lion king
   e. easter bunny

3. 5 queries with keywords on the data that was not crawled in the domain
   a. Kangaroo
   b. honey bee
   c. zoo
   d. rat hole
   e. shark

4. 5 queries which are somehow related to animal but can be used in other domains
   a. forest
   b. poaching
   c. animal rescue
   d. pork
   e. unicorn

The results obtained were not very promising and universal set of weights could not be arrived. Following were some of the findings:

1. Some queries like, kangaroo and hippopotamus could not fetch any documents at all, which means the data we crawled did not contain neither the pages nor any reference to these words. In such cases, relevance model did not work

2. Some queries fetched very relevant documents like, elephant and pork.

3. Some queries fetched partial results, like, Easter bunny. This query gave most results related to Easter but very few results related to bunny and mostly no result, which include both.

Some issues realized:

1. Performance issues as too much time was taken for some queries to get the stabilized weights, hence could not be performed on the fly
2. Each query needed its own unique set of weights

Queries produced by the algorithm:

Bird cage → bird cage rough extant fee amp

Parrot → parrot mealy special

Pseudo Relevance feedback:

Another query expansion technique is implemented which is query expansion through local clustering. It operates solely on the documents retrieved but this is not suitable for web searches because it is time consuming. There are three strategies of building local clusters:

•Association clusters

– Consider the co-occurrence of stems (terms) inside documents

• Metric Clusters

– Consider the distance between two terms in a document

Query Expansion – Srabonti Chakraborty, sxc154030

• Scalar Clusters

– Consider the neighborhoods of two terms

We implemented metric clustering algorithm. The queries tested with each of the clustering techniques were determined by:

- Queries with only one keyword that is in the relevant (i.e.) animal domain
- Queries with two or three keywords in animal domain
- Queries with stop words, ambiguous or irrelevant words

We are going to explain the results for input query: "elephant"

The following is the list of local vocabulary is:

elephant, wikidata, wikidata, jump, navigation, search, mammal, edit, language, label, description, known, english, mammal, statements, instance, common, name, references, subclass, elephantidae, image, african, bush, elephant, jpg, african, african, forest, elephant, elephant, jpg, jpg, references, references, elephas, maximus, bandipur, jpg, jpg, wildlife, wildlife, taxon, source, ivory, references, references, unicode, character, references, references, commons, category, elephantidae, elephantidae, reference, imported, from, from, german, wikipedia, topic, main, category, category, category, category, elephants, references, references, said, same, conservation, alerts …

The list of local stems are:

elephant, wikidata, mammal, references, african, jpg, elephantidae category, wikipedia, freebase, reserve, each, elephant, reserve, ranger, questions, wildlife, wwf, privacy, policy, policy, privacy, policy, has, changed, continuing, use, site, agreeing, new, policy, privacy, policy, privacy, site, help, world, wildlife, fund, elephant, home, search, new, coloring, index, coloring, pages, elephants, coloring, pages, coloring, pages …

Query Expansion – Srabonti Chakraborty, sxc154030

Original Query	Expanded Query
elephant	elephant africa asia
tiger	tiger species sumatran
dolphin	dolphin risso bottlenose
squirrel	squirrel palm monkey
cuckoo nest	cuckoo nest india omnivorous outreach
river fish	river fish thank tam otten
tiger cat	tiger cat breed fishermen wrong amur
honey bee	honey bee eaten bumbari specialize weren
pork	pork industry
forest	forest elephant africa
zoo	zoo diego san
lion king	lion king crab pencil sculptolithode tallest
kennel	kennel american club
zoo keeper	zoo keeper diego taking animal
poaching	poaching elephant crisis
Chihuahua	Chihuahua article animal
rat hole	rat hole women maze ration giant
shark	shark western whiskers
hippo	hippo hippopotamidae common
penguin	penguin gentoo humboldt

The result of the queries were mostly relevant when those are expanded. Some of the queries like "kangaroo" and "poaching" was not retuning any result when searched as the original query but the expanded query returned many relevant results. We have used the first 10 documents for getting the local vocabulary.

We realized that the result of the query is very much dependent on the amount of data crawled and the ranked results. For "shark" the top results were, "Cookie cutter shark", "Luminous shark", "Blue shark", "Nurse shark/ Oceana". These are all different varieties of shark. Again, for "cuckoo nest" all the results were about cuckoo. Whereas, the expanded result of "penguin" is "penguin gentoo humboldt" which is very relevant with penguin but a very partial result.

Collaboration to prepare the user interface:

The API calls were made to run the query expansion.
(in QueryExpansionServlethandler.java and APICalls.java)
1. The input for the API call were the initial query and SOLR output.
2. After the query expansion, the expanded query was returned as an output to the API call.