**The University of Texas at Dallas**

**CS 6322**

**Information Retrieval**

**Fall 2016**

**Class Project Report**

***Search engine for Latest Electronic Products***

**TEAM 7**

**Team Members:**

**Javnika Devendra Sharma (jxs153030)**

**Karthika Karunakaran (kxk152430)**

**Niraj Gadakari (n)**

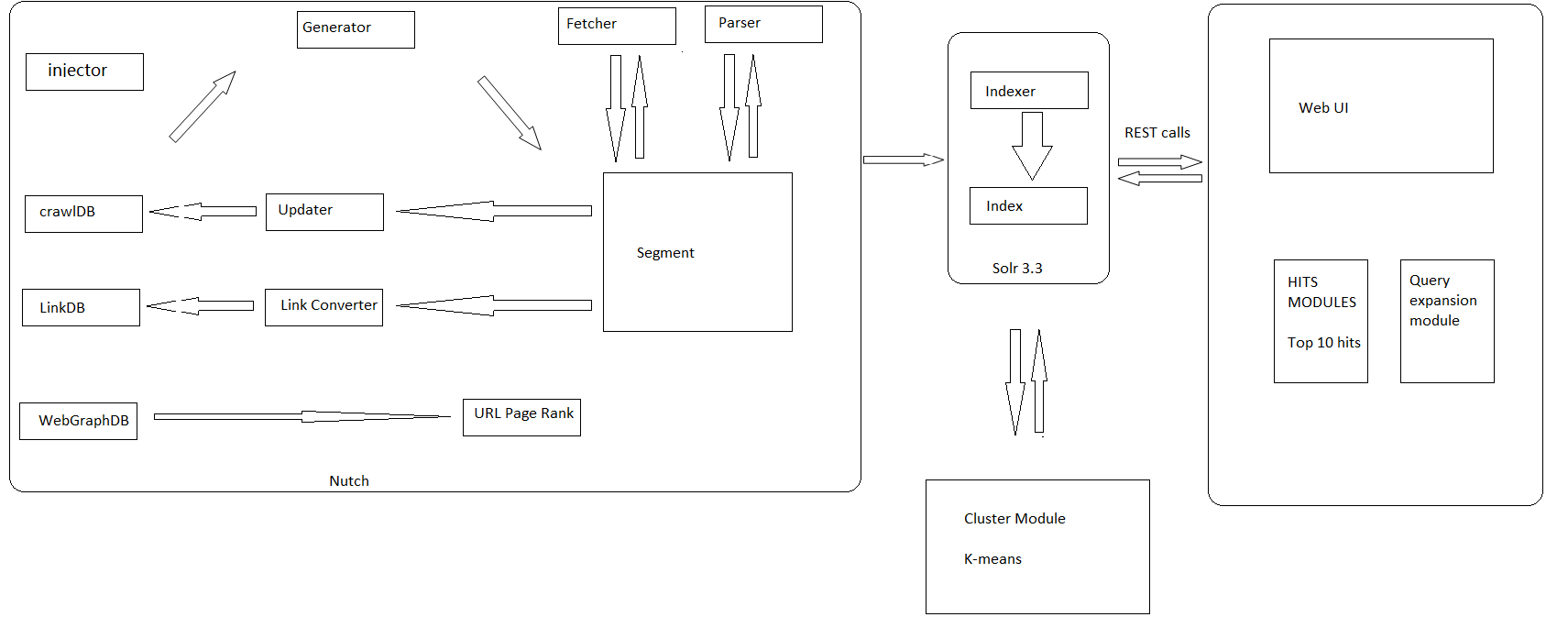
**Sharannya Ranjith (sxr154730)**

**Yu Wang (yxw124430)**

1. **The Problem: Generate a Search Engine for latest Electronic Products**

The objective of this project is to implement a search engine for the latest Electronic Products. The architectural diagram followed for the development of this search engine is given below-

**Architectural Diagram**



The responsibilities handled by each team member are:

* **Crawling:** Javnika Devendra Sharma
* **Indexing:** Karthika Karunakan
* **User Interface:** Sharannya Ranjith
* **Clustering:** Niraj Gadakari
* **Query Expansion:** Yu Wang

This has been a positive learning experience where all of us had the opportunity to relate our academic knowledge and to put into practice all the theory learnt in the class room. It was the perfect opportunity to obtain hands-on experience in developing a search engine. We got to learn a lot and experience the dynamics of working with a team

We faced an initial hitch where we started crawling with Nutch but it was incompatible with Lucene API used for Indexing and we it was resolved as we decided to go ahead with Crawler4j

Being a part of a team can sometimes turn out to be difficult especially if some people have different opinions and trying to reach a consensus is difficult in such cases. Handling any problem in the team is possible only when we communicate with each other. And this is what we urged everyone in the team to do as well. Once we hear the other person’s point of view we can understand each other better and this automatically resolves any conflicts. Apart from implementing our own parts of the search engine, we helped one another in their sections as well.

2. Crawling Javnika Devendra Sharma(jxs153030)

1. The steps followed to achieve crawling are –
2. Provide the list of seed URLs to crawl,how many pages needed to be crawled and directory in which result are to be saved.
3. Pages that pass the filtering process,are then parsed.
4. The required information extracted are then written to file which in turn are used by indexer.
5. Add URLs to queue while taking care of duplicate URLs.
6. Repeat the above steps till the queue is empty.
7. The software used for crawling is Open Source Java library Crawler4j
8. The Seed URLs that were used for crawling are –
9. <http://www.samsung.com/us/>
10. <http://www.lg.com/us>
11. <http://shop.panasonic.com/>
12. <http://www.htc.com/us/>
13. <http://www.ebay.com/rpp/electronics>
14. <https://waycoolgadgets.com/category/electronics>
15. <http://www.popsugar.com/tech/>
16. <http://www.newelectronics.co.uk/electronics/>
17. <http://www.electronicsweekly.com/news/products/>
18. <https://www.thegrommet.com/tech-gadgets>
19. <http://thegadgetflow.com/portfolio/category/all-things-tech/>
20. The total number of pages crawled are 112687.
21. Size of the crawled data is 3.4 GB
22. This data is stored in the form of text files as a JSON object which mainly contains URL,title, page information, outgoing link form page, incoming links from page,total outgoing link. This way I provided hyperlink information for the Karthika Karunakaran that generated the index and relevance models.
23. In order to avoid duplicates URLs in the crawl, the crawler normalizes new URLs to remove duplicates.

1. Challenges Faced : Javnika Devendra Sharma(jxs153030)
2. During crawling, we had initially started with the software Nutch but were facing a lot of issues as it was incompatible with our version of Lucene which was used for indexing. So we switched to Crawler4j, which was effective in solving our problem.
3. Too many sub­ categories in travel to search. (eg television, mobile, refrigerator, etc)
4. Crawling was taking too much time.
5. Space and time complexity was always a major issue (Huge dataset ).

3. Indexing and Relevance Model Karthika Karunakaran (kxk152430)

Tools Used**:** Spring Boot 3.8.2, Lucene APIs 4.8, Maven, Jung APIs and few other libraries to create a web service in java

As a result of crawling, the data along with web graph in the form of adjacency matrix, page url, title, content details were generated. I merged the collected data with multi-threads into one.

Indexing :

I parsed the data into required Document format which is based on Apache Lucene, using the Lucene API the indexing these documents was performed. The indexed binary files are created and stored.

The index contains the following information about a term

i. Field Name

ii. Term Dictionary

iii. Term Frequency

iv. Term Proximity data

v. Normalization factors

vi. Term Vectors

Relevance models:

The relevance model used is vector space model. The class TFIDF Similarity in Lucene is extended to calculate the weight involving normalization and get similarity. Top 10 results are returned from the IndexSearcher from the collection. Both Hits and Pageranking are combined with this model for results. Duplicate filter is called for a sanity check.

Statistics of the webgraph:

* Total number of nodes : 112687
* Maximum number of outgoing links : 502

Pageranking:

PageRanking Algorithm is implemented using the Jung APIs to rank the pages which are obtained from the webgraph with incoming links and outgoing links.

All the APIs of jung are -

JUNG-algorithms-2.0.jar

Jung-api 2.0.1.jar

Junggraph-impl 2.1.jar

Karthika Karunakaran (kxk152430)

Steps of PageRanking:

1. Using the inlinks and outlinks information of all the given links we build a DirectedSparseGraph.

2. From this graph pageranking is calculated with the dampling factor and total no of url using the Pagerank formula.

3. This pagerank is sorted to get the top ranked and relevant links.

4. It is also included in the indexed information document before indexing.

5. The top 10 results based on the pagerank is returned for the input query.

Hits:

The Lucene API collects and returns the top pages based on hit score calculated.

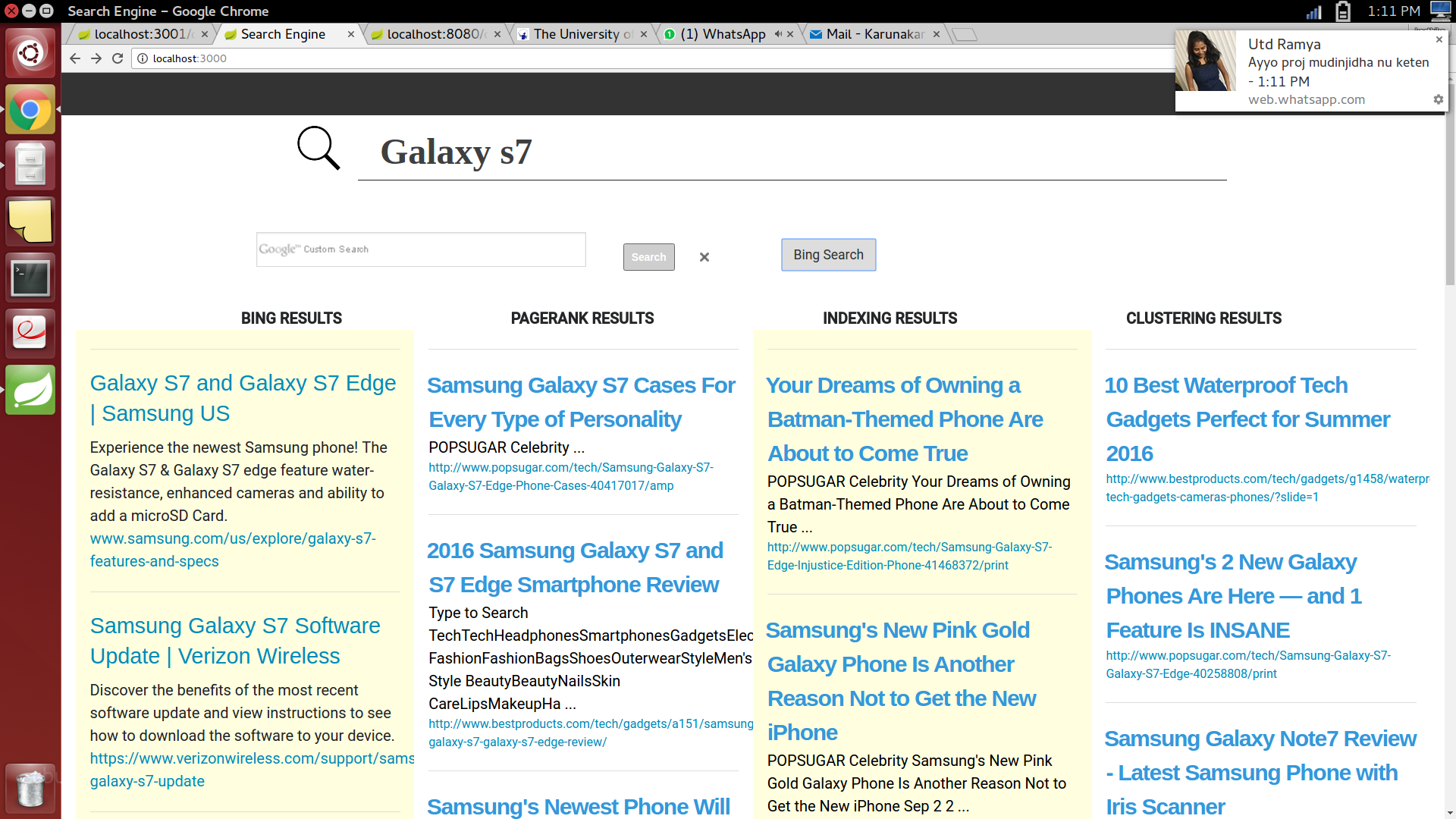
Collaborating it with other team members:

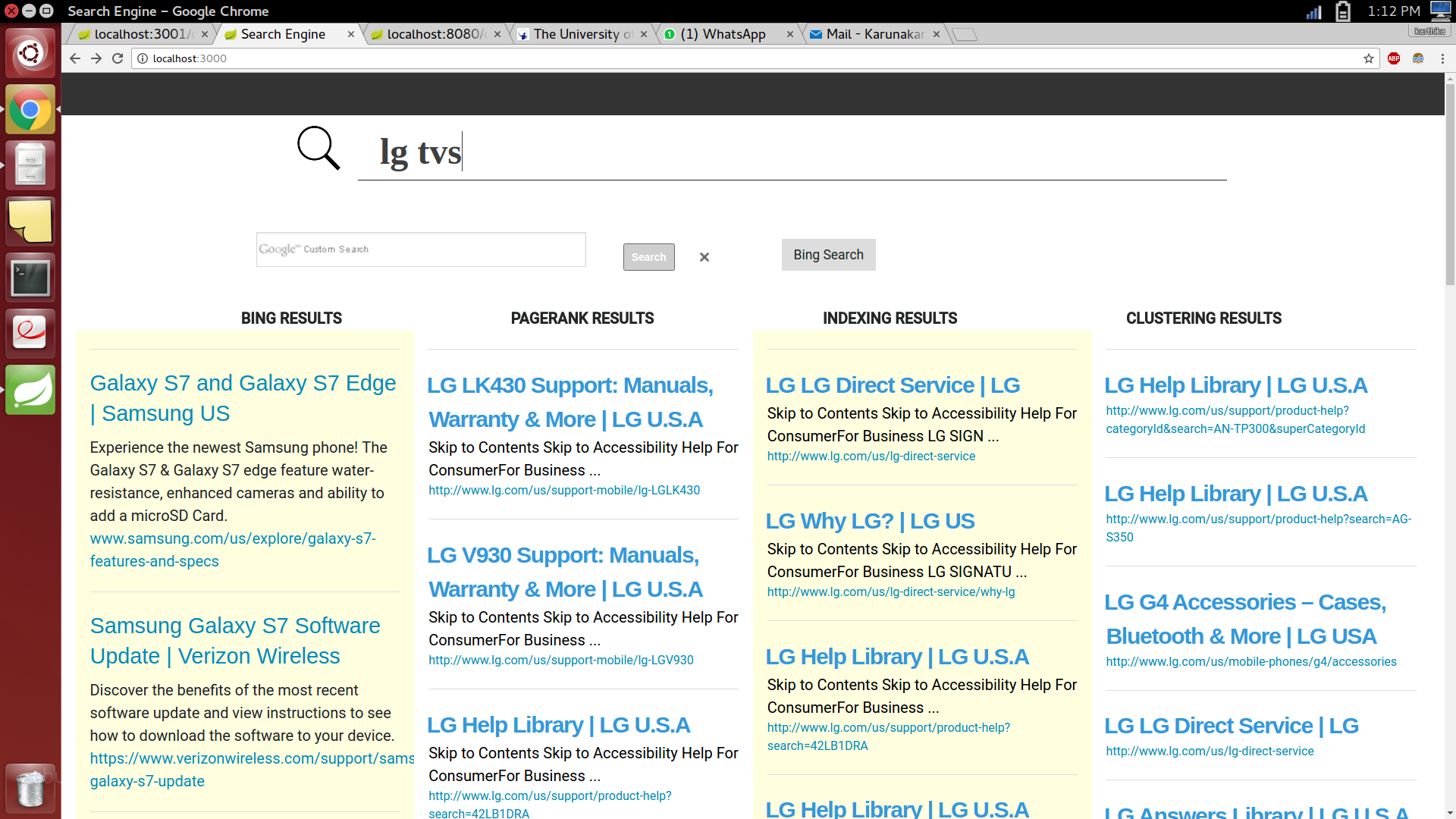
I worked closely with Sharanya to pull my results on the UI and compare with Bing and Google results, we had worked together and decided on how to return the results, how the URLs are displayed up and place juxtaposing to google and bing. We decided to have Restful webservices to send request from the UI. I have used 20 queries to check the relevance results.

I also worked with Javnika to make sure relevance of docs crawled is close otherwise suggesting her to crawl with different seeds.

I work with Niraj and Yu as well to lend my segment results for further clustering and query expansion. After clustering the results were pulled to UI similar to search results and compared with page ranking and hits results. The indexed segments are given to him for clustering to avoid out of memory error. By comparing results of clustering thus shown in the results.

Below are the screenshots of the PageRank and Hits results.





4. User Interface Sharannya Ranjith (sxr154730)

The user interface has been designed using these technologies:

1. **Spring 3.8.2** Spring was used to set up the REST calls to the server. The Spring Framework provides a comprehensive programming and configuration model for modern Java based enterprise applications on any kind of deployment platform. A key element of Spring is infrastructural support at the application level: Spring allows teams to focus on application level business logic, without unnecessary ties to specific deployment environments. Tomcat, which was built in Spring was used as the application server. Apache Tomcat software is an open source implementation of the Java Servlet, JavaServerPages, Java Expression Language and Java WebSocket technologies. (<https://projects.spring.io/springframework/>)
2. **Apache Maven:** Apache Maven was used to build the project. Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information. (<https://maven.apache.org/>)
3. **jQuery 2.0.2** The jQuery framework is used to make the AJAX calls to the RequestMapping in the Controller. jQuery is a fast, small, and feature rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy to use API that works across a multitude of browsers. (https://jquery.com/download/)
4. **Bootstrap:** Bootstrap was used as the CSS framework. Bootstrap (frontend framework) is a free and open source frontend library for creating websites and web applications. It contains HTML and CSS based design templates for forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. (http://getbootstrap.com/2.3.2/)
5. **Google Custom Search Engine:** To populate the Google search results I have created a Google Custom Search Engine. (<https://developers.google.com/web-search/docs>)
6. **Bing Search Engine:** Used a jQuery plugin to populate the Bing results. (<http://cbenard.github.io/jquerybingsearch/>)

**Collaborating with the indexing teammate:** Once the documents were crawled and indexed, I retrieved the results by using an REST API and fired queries accordingly to retrieve the documents and returned the results to the UI.

**Example of generating the results in the UI:** jQuery makes an AJAX call to the Rest Controller in the spring application, deployed on the tomcat server. The request controller receives the request and fires the query to the Indexing API depending on what query was fired and returns the results, the raw results are then processed and put into an object which is sent back to the UI as a JSON object.

Sharannya Ranjith (sxr154730)

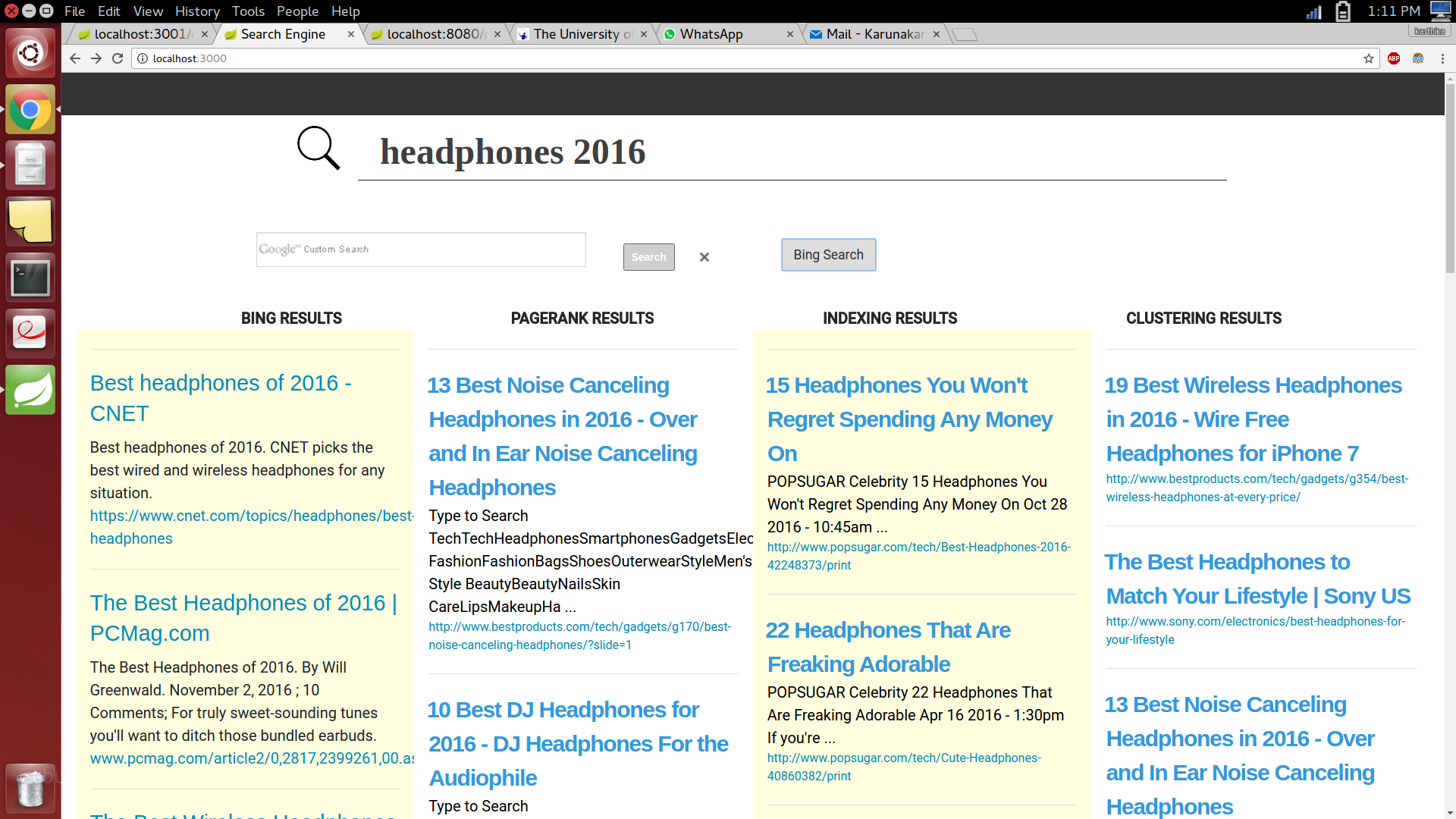
**How the queries were tested on the search engine:** We generated a random number of electronic product queries and tested about 50 queries. The queries show up in the autosuggest list on the UI. The results were very good on all except for page ranking, as the links with maximum inlinks have higher score and some website were more prominent with the page ranking score. After the relevance model was built, we further executed 50 queries to test the validity of our model.

**Collaborating with the clustering teammate:** Once the URLs and the related clusters were given to me by the clustering teammate, I using an REST API and fired queries accordingly to retrieve the documents. The results were presented in the UI.

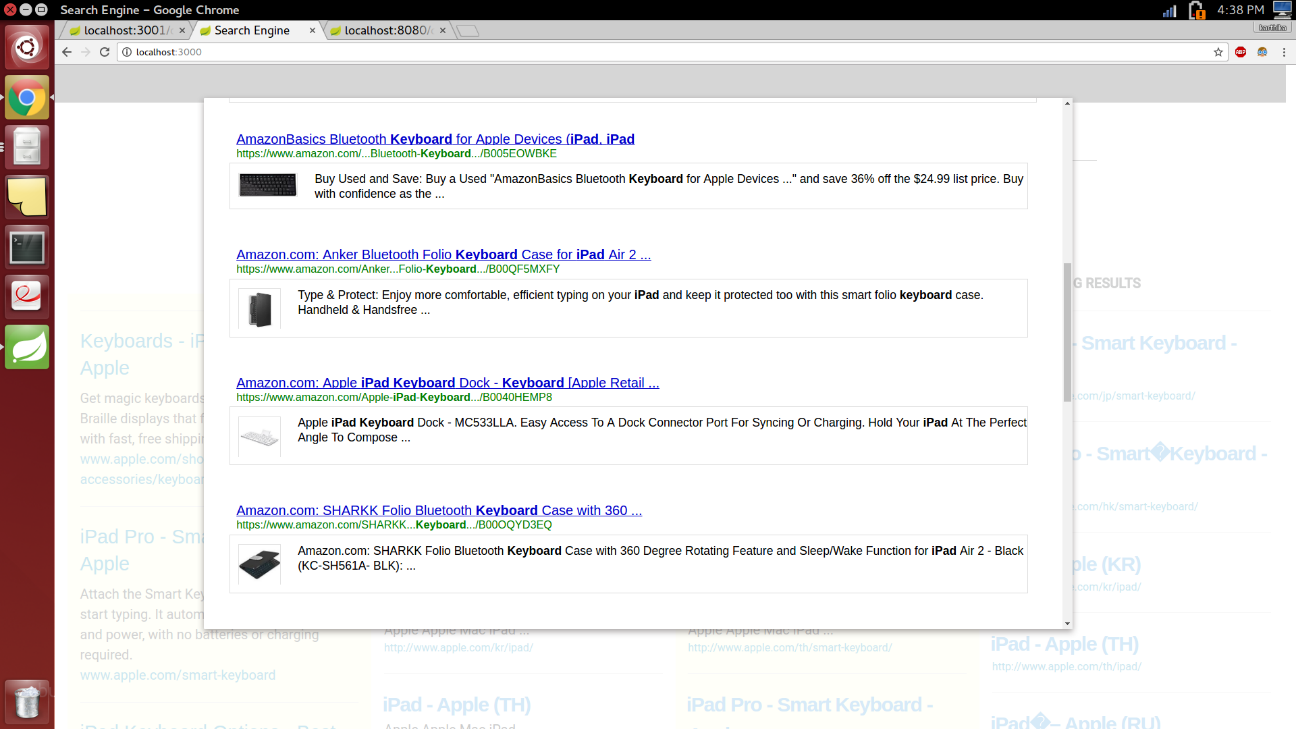
**How our search engine compares to Google and Bing:** Our search engine performs pretty well when compared to both. We can see some common results from common website in our search engine and google. It depends on the documents crawled.

Below are the screenshots of the results from Google and Bing compared with our results.

BING RESULTS AND OUR RESULTS



GOOGLE RESULTS



**Three examples of the queries and the results produced by your search engine, as well as the results of Google and Bing:**

|  |  |  |  |
| --- | --- | --- | --- |
| QUERY | GOOGLE | BING | OUR RESULTS |
| Phone LG | http://www.lg.com/us/cell-phones | www.***lg***.com/us/cell-***phones*** | http://www.popsugar.com/tech/LG-Shows-Off-Candybar-Shine-Phone-209682 |
|  | http://www.lg.com/us/t-mobile-phones | www.***lg***.com/us/mobile | http://www.popsugar.com/tech/Have-You-Ever-Taken-Artistic-Picture-Your-Cell-Phone-272466 |
|  | http://www.lg.com/us/att-phones | https://www.walmart.com/c/kp/***lg***-***phones*** | http://www.lg.com/us/home-video/lg-BD650-blu-ray-dvd-player |
| Galaxy S7 | http://www.samsung.com/us/explore/galaxy-s7-features-and-specs/ | **www.samsung.com**/us/explore/**galaxy-s7**-features-and-specs | http://www.popsugar.com/tech/Samsung-Galaxy-S7-Active-41566768/print |
|  | http://www.gsmarena.com/samsung\_galaxy\_s7-7821.php | https://***www.cnet.com***/products/samsung-***galaxy-s7*** | http://www.popsugar.com/tech/Samsung-Galaxy-S7-Galaxy-S7-Edge-Phone-Cases-40417017/print |
|  | https://www.verizonwireless.com/smartphones/samsung-galaxy-s7/ | https://***www.att.com***/cellphones/samsung/***galaxy-s7***.html | http://www.bestproducts.com/tech/gadgets/news/a140/samsung-galaxy-s7-and-s7-edge-reviews/ |
| Fitbit | https://www.**fitbit**.com/ | www.***fitbit***.com | http://www.bestproducts.com/tech/gadgets/gmp1585/fitbit-reviews/ |
|  | https://www.amazon.com/Fitbit-Wireless-Activity-Sleep-Wristband/dp/B00BGO0Q9O | www.***bestproducts***.com/tech/gadgets/g1585/***fitbit***-reviews | http://www.bestproducts.com/tech/gadgets/g1585/fitbit-reviews/?slide=7 |
|  | https://play.google.com/store/apps/details?id=com.fitbit.FitbitMobile&hl=en | http://www.bestbuy.com/site/brands/fitbit-health-fitness-products/pcmcat331400050000.c?id=pcmcat331400050000 | http://www.bestproducts.com/tech/gadgets/g1585/fitbit-reviews/?slide=8 |

**Challenges faced:**

1. We faced a problem when we tried incorporating Servlets in Spring Framework
2. Google’s Custom Search API allows Google search only to happen by typing the query explicitly in the search bar. Hence we were unable to incorporate the functionality in the UI but implemented it as a popup.

5. Clustering Niraj Gadakari (nxg152930)

Document clustering (or text clustering) is the application of cluster analysis to textual documents.

API’s and Libraries used:

SciKit-learn: from sklearn.cluster import K-Means

from scipy.cluster.hierarchy import ward, dendrogram

Carrot2: Carrot2 is an Open Source Search Results Clustering Engine. It can automatically organize collections of documents into thematic categories.

The following algorithms were implemented and experimented for over 500 queries.

Flat Clustering:

K-means:

K­Means clustering is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining. k­means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells.

I experimented with different values of k for different sets of documents. I found out that the k-means gave good clustering results with k = 8. Here the number of documents used were 12000.

However, as I increased the data files, I was running into memory exception.

The documents were clustered by topics using a bag-of-words approach. A scipy.sparse matrix was used to store the features instead of standard numpy arrays.

The following feature extraction method were used for clustering algorithm:

TfidfVectorizer:​It uses an in-­memory vocabulary (a python dict) to map the most frequent words to features indices and hence compute a word occurrence frequency (sparse) matrix. The word frequencies are then reweighted using the Inverse Document Frequency (IDF) vector collected feature­wise over the corpus.

Hierarchical Clustering:

Ward’s Agglomerative clustering: Ward's method starts out with n clusters of size 1 and continues until all the observations are included into one cluster. Ward clustering is based on a Feature matrix. It recursively merges the pair of clusters that minimally increases within-cluster variance.

Studied the behavior for different query results. Plotted the dendrogram to analyze the cluster results. The algorithm worked fine with fewer documents, but I ran into memory issues with large documents. Also, the algorithm lacked in labelling the cluster results which would be useful to the user.

Niraj Gadakari (nxg152930)

On further research, I found another API provided by Carrot2. I decided to use an algorithm called LINGO, which smartly uses the indexing results from Apache Lucene for all the 100k+ documents, and organizes the clustering results into thematic categories based on the search query.

Flat Clustering:

LINGO Clustering:

When designing our web clustering algorithm, special attention was paid to ensuring that both contents and description (labels) of the resulting groups were meaningful to the users. The majority of currently used text clustering algorithms follow a scheme where cluster content discovery is performed first, and then, based on the content, the labels are determined. Unfortunately, this may result in some groups' descriptions being meaningless to the users, which in turn, is very often caused by the nonsensical content of the clusters themselves. To avoid such problems, we used LINGO clustering algorithm, which adopts a radically different approach to finding and describing groups.

The general idea behind LINGO is to first find meaningful descriptions of clusters, and then, based on the descriptions, determine their content.

This approach yielded better results, and the cluster description were meaningful, and relevant.

I used the results of clustering to improve the search results. Specifically, I incorporated the scores given to the resulted clusters by the Carrot2 API for a particular query. Thus it improved our results. And the results were displayed with each cluster having meaningful cluster description.

The cluster-scores were calculated based on the following formula:

cluster-scores = label-scores \* member-count

More than 500 queries were used to observe the cluster results. Based on the cluster results, I discussed with the person crawling the data to include some more relevant seeds to crawl.

Niraj Gadakari (nxg152930)

As our search engine is about Latest Electronic Products, I have selected the following queries for demonstration:

Search Query = “iPhone 6 plus”

Created 24 clusters

The following clusters were formed with description for each cluster. Each cluster has the number of documents specified in the description.

iPhone 6 and 6s Cases (22 docs, score: 38.6)

Screen Protectors (16 docs, score: 50.29)

iPhone 6 Plus Cases and Covers (13 docs, score: 23.46)

Protective iPhone Screen Covers (10 docs, score: 24.7)

Apple (8 docs, score: 22.81

Adhesive and Glass Screen Protectors (6 docs, score: 43.15)

PM\_Logo (5 docs, score: 27.2)

Protective Cases for Your iPhone (5 docs, score: 2.15)

Battery (3 docs, score: 18.55)

Love (3 docs, score: 24.32)

iPhone Accessories for Your New iPhone (3 docs, score: 22.96)

Apple Finally Announces a Repair Program (2 docs, score: 21.31)

New Apple (2 docs, score: 21)

Pixar iPhone Cases (2 docs, score: 3.22)

Pretty Phone Cases That Will Induce Envy (2 docs, score: 12.77)

Samsung Cases for You and the BFF (2 docs, score: 13.88)

Waterproof (2 docs, score: 18.09)

iPhone Cases POPSUGAR Tech (2 docs, score: 9.87)

Other Topics (17 docs, score: 0)

Niraj Gadakari (nxg152930)

Search Query = “iPhone 6 plus”

Created 25 clusters

The following clusters were formed with description for each cluster. Each cluster has the number of documents specified in the description.

Cheap Yet Quality Headphones and Earbuds  (24 docs, score: 29.97)

Wireless Headphones  (21 docs, score: 23.43)

Sony  (14 docs, score: 24)

Stylish On Ear Headphones  (12 docs, score: 40.53)

Apple  (11 docs, score: 15.75)

Bluetooth Headphones  (11 docs, score: 30.58)

Noise Canceling  (10 docs, score: 49.51)

Headphones Review  (9 docs, score: 30.92)

Earbuds and Sports Clip Headphones Panasonic  (6 docs, score: 19.48)

Apple UK  (5 docs, score: 22.08)

Headphones Speakers  (5 docs, score: 0)

PM\_Logo  (4 docs, score: 22.61)

Bose Earbuds and Wireless Headphone Reviews  (3 docs, score: 21.13)

Cheap Bluetooth Headphones 2016  (3 docs, score: 10.71)

DJ Headphones for the Audiophile  (3 docs, score: 33.17)

Cheap Over Ear Headphones  (2 docs, score: 5.73)

Headphones Compatible with Your MacBook Air  (2 docs, score: 25.48)

Headphones That Are Freaking Adorable  (2 docs, score: 24.78)

Headphones You Won't Regret Spending Any Money  (2 docs, score: 10.16)

Reviews of Beats Audio Headphones  (2 docs, score: 16.56)

Unveils  (2 docs, score: 12.73)

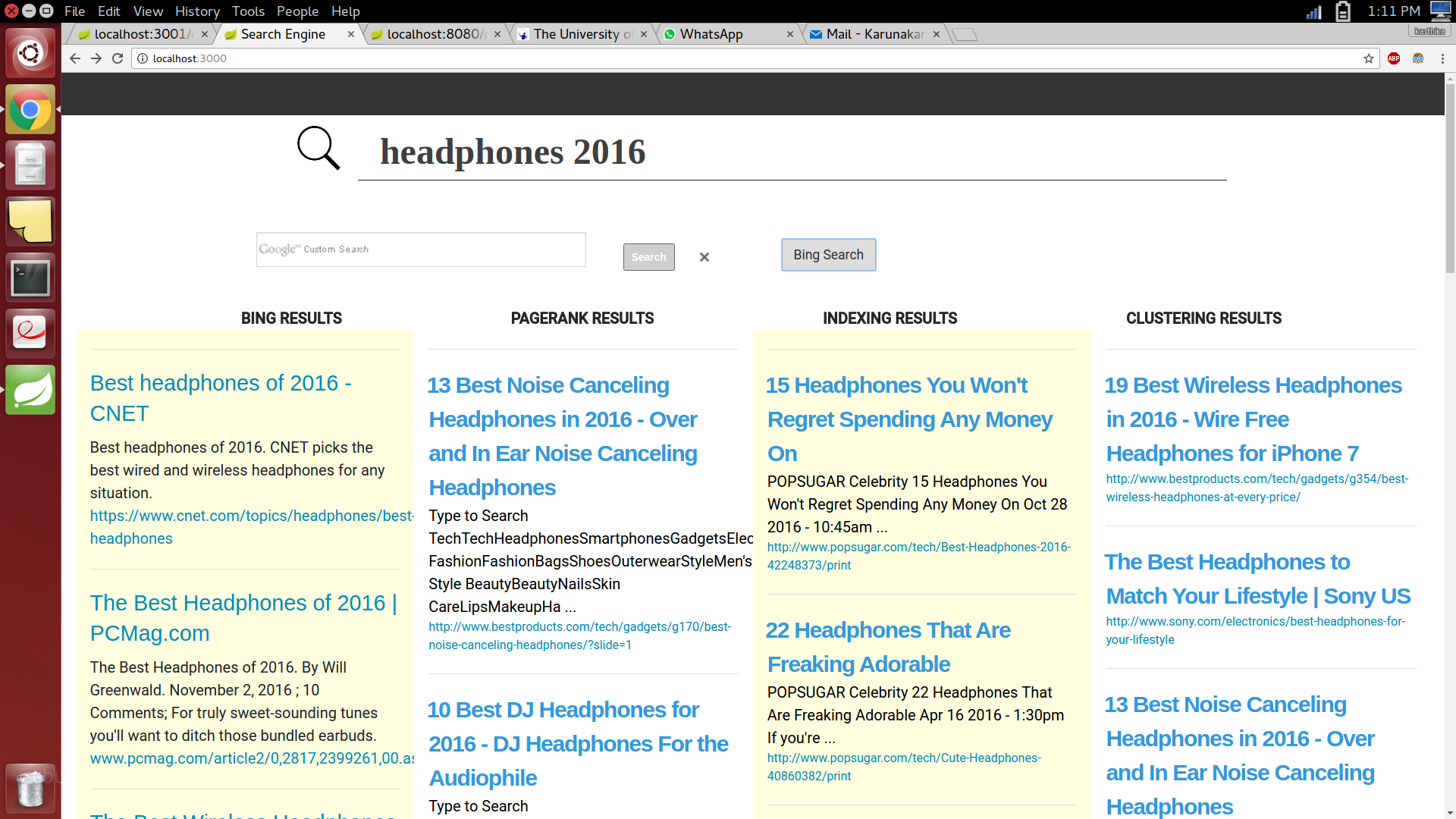
Workout  (2 docs, score: 13.27)

Other Topics  (5 docs, score: 0)

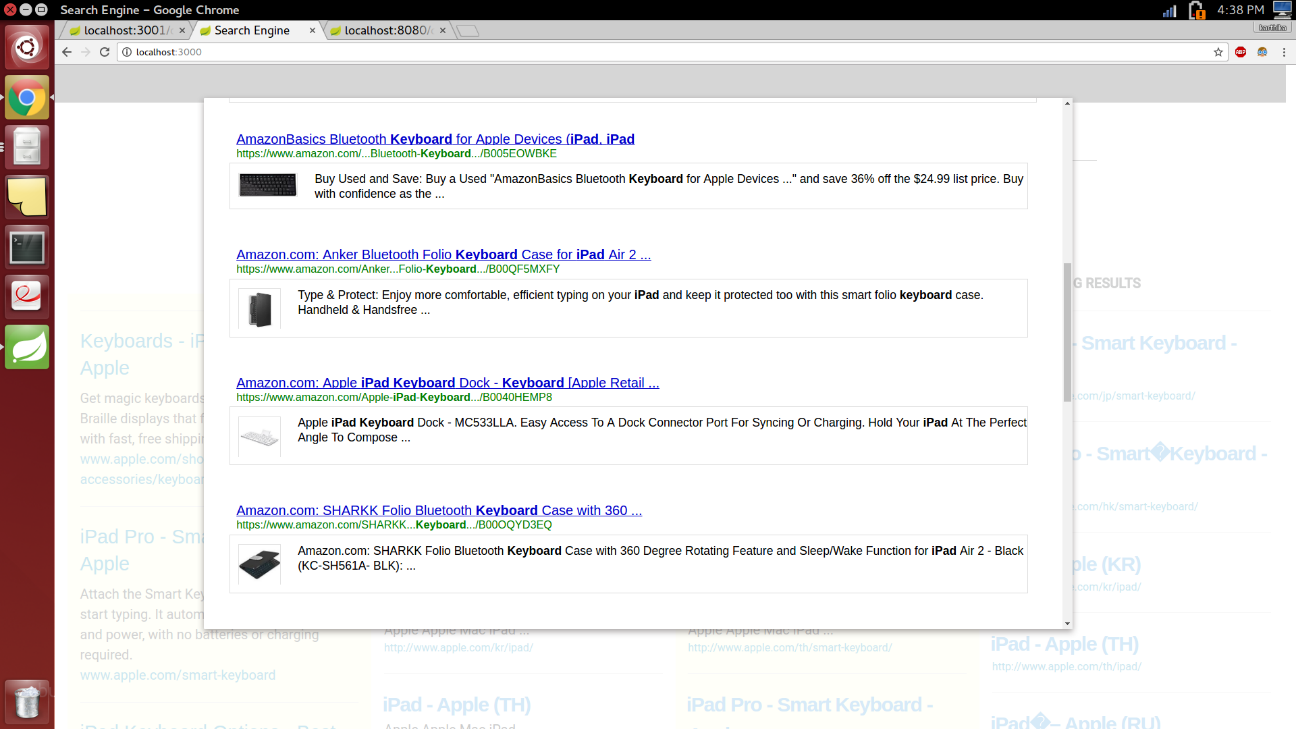
Niraj Gadakari (nxg152930)

Below are the screenshots of the results from Google and Bing compared with our results.

BING RESULTS AND OUR RESULTS



GOOGLE RESULTS



6. Query Expansion Yu Wang (yxw124430)

Query expansion is the process of reformulating a seed query to improve retrieval performance in information retrieval operations. In the context of web search engines, query expansion involves evaluating a user's input (what words were typed into the search query area, and sometimes other types of data) and expanding the search query to match additional documents.

Query expansion involves techniques such as:

1. Finding synonyms of words, and searching for the synonyms as well.
2. Finding all the various morphological forms of words by stemming each word in the search query.
3. Fixing spelling errors and automatically searching for the corrected form or suggesting it in the results

Reweighting the terms in the original query

Relevance feedback is to take the results that are initially returned from a given query and to use information about whether or not those results are relevant to perform a new query.

We implemented Rocchio Algorithm as follows:

**Rocchio Algorithm**

1. Retrieve the top 10 documents which get the most match for the query

2. Calculate the query vector and document vector separately based on their term

tf-idf value



3. Further calculate query vector and document vector with Rocchio algorithm

4. Combine both query and document vector and sort the terms based on its weight in descending order and take the first 6 as the expanded query. For the Cluster models, the correlation factor between query and document terms were sorted in descending order and the top 6 document terms were picked up for expanded query outcome

**Associate cluster**





Yu Wang (yxw124430)

**Metric cluster**





**Scalar cluster (with metric cluster vector)**



Original Query: Electronic

Expanded Query: Information Products Components Company Computer Center

Example expanded query on server console and the user UI



7. Discussion

Every member of the team was present and ever keen to discuss the implementation ideas before starting every module. Each team member had inputs to efficiently implement a particular module. During crawling, we had initially started with the software Nutch but were facing a lot of issues as it was incompatible with our version of Lucene which was used for indexing. During that time, one of the teammate gave the suggestion to go ahead with Crawler4j, which was effective in solving our problem.

8. Conclusion

We believe that we did our best in implementing this project- A search engine for the latest electronic products. We were able to improve existing results and obtain more relevant ones. The further enhancements to this project can be done optimizing the algorithms and by improving time and space complexities.