Project Report CS 553 Fall 2015

Overview

In this project we have built a search engine. This search engine could be used by a user of the internet to search for web pages he/she is looking for. The user can provide words associated with the web page he/she is looking for as the input, and the search engine looks for the words entered by the user in the corpus (collection of the parsed data of web pages that the search engine has) and returns the most appropriate results (URLs of web-pages which contain the word(s) provided as input) ranked using the Ranking Algorithm explained in the report. The project is divided in three phases which are explained in the project later.

This report is organized as follows -

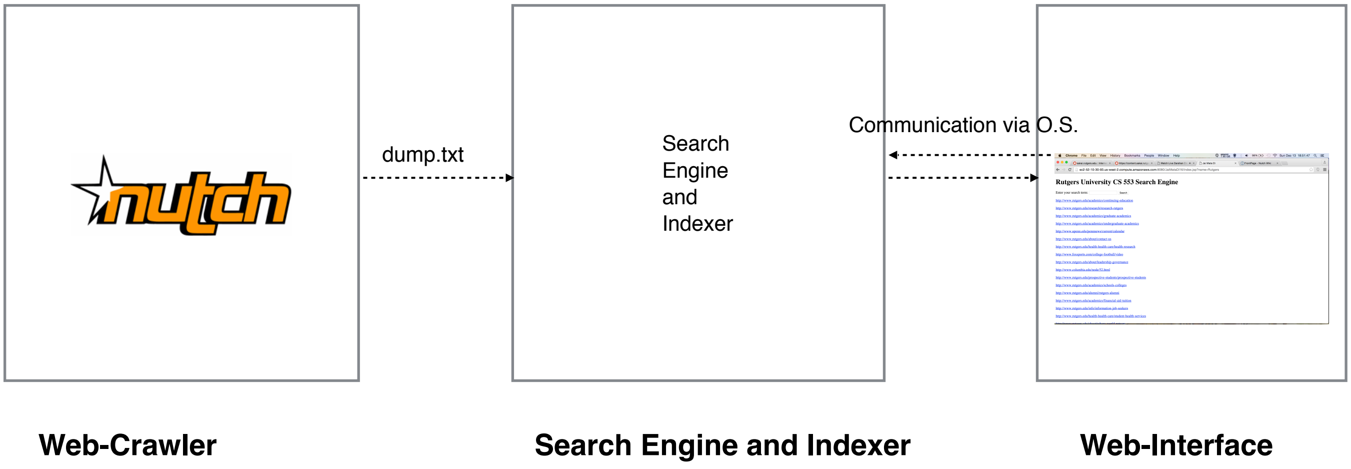
Overview of Phases

This project was divided in 3 phases, each of which maps to a specific module of the search engine -

1. Phase 1 - nutch a web-crawler was used to crawl the internet. It accepted seed lists consisting of educational websites and sports websites. The output of this phase was dump.txt file which had 76.2 MB size and contained more than 1.6 million lines containing all the fetched web pages, urls, and metadata.

2. Phase 2 - This phase accepts as input the dump.txt file generated as output to phase 1. It then parses the dump.txt file extracts all the parsed data of the fetched web-pages and indexes all the web-pages. The index is an inverted-index (which means that it is a key-value index where the key is the word and the value is the document number which contains the word). In this phase a Ranking Algorithm was implemented to rank the outputs. The input to the Indexer and Search Engine is the search term (consisting of words being looked for) and output is the ranked list of URLs of web-pages containing that word.

3. Phase 3 - Web-Interfacing. A Web-Interface (website) which allows the users to enter the search term, which is accepted by the Indexer and Search Engine (described in Phase 2) as the input, and then subsequently displays the output from the Indexer and Search Engine on the web-interface. The interface exposes the search engine to a user of the internet, making it possible for a common user to use it.


nutch
dump.txt
Search Engine and Indexer
Communication via O.S.
Chrome File Edit View History Bookmarks People Window Help
Sun Dec 13 18:51:47
sakai.rutgers.edu : Interne
https://content.sakai.rutge
Watch Live Darshan D
Jai Mata Di
FrontPage - Nutch Wiki
ec2-52-10-30-93.us-west-2.compute.amazonaws.com:8080/JaiMataDi16/index.jsp?name=Rutgers
Search
Web-Crawler
Search Engine and Indexer
Web-Interface

Rutgers University CS 553 Search Engine

Enter your search term:

http://www.rutgers.edu/academics/continuing-education

http://www.rutgers.edu/research/research-rutgers

http://www.rutgers.edu/academics/graduate-academics

http://www.rutgers.edu/academics/undergraduate-academics

http://www.upenn.edu/pennnews/current/calendar

http://www.rutgers.edu/about/contact-us

http://www.rutgers.edu/health-health-care/health-research

http://www.foxsports.com/college-football/video

http://www.rutgers.edu/about/leadership-governance

http://www.columbia.edu/node/52.html

http://www.rutgers.edu/prospective-students/prospective-students

http://www.rutgers.edu/academics/schools-colleges

http://www.rutgers.edu/alumni/rutgers-alumni

http://www.rutgers.edu/academics/financial-aid-tuition

http://www.rutgers.edu/info/information-job-seekers

http://www.rutgers.edu/health-health-care/student-health-services

Phase 1 - Crawling using Nutch

In Phase 1 we used Nutch 1.0 crawler to crawl the web. Apache Nutch is a highly extensible and scalable open source web crawler software. The *input* for this phase were 2 seed lists. Seed lists consisted of the urls, these urls act as starting points for crawling. We used the nutch tutorial for this purpose.

The internet can be compared to a giant graph where each of the computers connected to it, is like a node of the graph. Since nutch by default crawls in Breadth-First-Search manner, therefore it was natural to provide a seed list. This is because BFS is a graph traversal algorithm which accepts the source node as input for it to begin traversing the graph. BFS begins traversal from the source node provided to it as the input. There is a second reason to provide the seed list - if specific websites need to be crawled then providing their exact urls is useful. Crawling the web is like traversing the giant-graph called internet, its just that, while crawling whatever the crawler hits upon (e.g. a web page) it fetches that, so that it could be used for other purpose (e.g. input for a search engine) later on.

We provide here the list of educational websites and sports websites which constituted the seed lists

http://www.harvard.edu/
http://www.princeton.edu/
http://www.yale.edu/
http://www.columbia.edu/
http://www.caltech.edu/
http://web.mit.edu/
http://www.rutgers.edu/
http://www.stanford.edu/
http://www.uchicago.edu/
http://www.upenn.edu/
http://www.duke.edu/
http://www.dartmouth.edu/
http://www.northwestern.edu/
http://www.jhu.edu/
http://wustl.edu/
http://www.brown.edu/
http://www.rice.edu/
http://www.vanderbilt.edu/
http://nd.edu/
http://www.emory.edu/home/
http://www.asu.edu/
http://www.washington.edu/
http://www.foxsports.com/
http://www.nbcsports.com/
http://www.bbc.com/sport/0/
http://www.realmadrid.com/en
http://www.wimbledon.com/
http://www.usopen.org/index.html
http://www.sportstats.com
http://bleacherreport.com/nfl
http://www.nfl.com/
http://www.nblcanada.com/
http://www.ussoccer.com/
http://www.fifa.com
http://www.uefa.com
http://www.usbasket.com/
http://princetonyouthhockey.org/
https://www.nsaa.org/
http://www.isaa.org/

As mentioned in the Nutch Tutorial the crawler - Nutch was used to fetch 2 times after going through the following steps

1. Seeding
2. Fetching
3. Parsing
4. Updating the DB
5. Fetching 2nd time
6. Parsing 2nd time
7. Updating DB 2nd time
8. Inverting all links

All the above steps are explained in detail in the nutch tutorial. We present here the significant observations. We show the commands and discuss and analyze the output of the commands below.

1.1 Crawl Database Analysis and Discussion

Universities Websites- It was observed that 3733 links were there. We have given below the command and the output.

```
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/apache-nutch-1.10$ bin/nutch readdb crawl/crawldb/ -stats
CrawlDb statistics start: crawl/crawldb/
Statistics for CrawlDb: crawl/crawldb/
TOTAL urls:    3733
retry 0:    3732
retry 1:    1
min score:    0.0
avg score:    0.012907581
max score:    1.552
status 1 (db_unfetched):    3147
status 2 (db_fetched):    497
status 3 (db_gone):    6
status 4 (db_redir_temp):    13
status 5 (db_redir_perm):    70
CrawlDb statistics: done
```

Sports Websites - It was observed that 2907 links were there. We have given below the command and the output.

```
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/JaiMataDiSports/apache-nutch-1.10$ bin/nutch readdb crawl/crawldb/ -stats
CrawlDb statistics start: crawl/crawldb/
Statistics for CrawlDb: crawl/crawldb/
TOTAL urls:    2907
retry 0:    2902
retry 1:    5
min score:    0.0
avg score:    0.014617475
max score:    1.604
status 1 (db_unfetched):    2316
status 2 (db_fetched):    533
status 3 (db_gone):    2
status 4 (db_redir_temp):    48
status 5 (db_redir_perm):    8
CrawlDb statistics: done
```

Understanding the Output

The ReadDb command is used read the crawl dump.

EDUCATION WEBSITES

```
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/apache-nutch-1.10$ bin/nutch readdb crawl/crawldb/ -dump joutput
CrawlDb dump: starting
CrawlDb db: crawl/crawldb/
CrawlDb dump: done


jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/apache-nutch-1.10$ bin/nutch invertlinks crawl/linkdb -dir crawl/segments

LinkDb: starting at 2015-10-12 04:45:28
LinkDb: linkdb: crawl/linkdb
LinkDb: URL normalize: true
LinkDb: URL filter: true
LinkDb: external links will be ignored.
LinkDb: adding segment: file:/home/jaimatadi/Documents/JaiMataDi/apache-nutch-1.10/crawl/segments/20151012040815
LinkDb: adding segment: file:/home/jaimatadi/Documents/JaiMataDi/apache-nutch-1.10/crawl/segments/20151012040927
LinkDb: finished at 2015-10-12 04:45:32, elapsed: 00:00:04
```

SPORTS WEBSITES

```
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/JaiMataDiSports/apache-nutch-1.10$ bin/nutch readdb crawl/crawldb/ -dump joutput
CrawlDb dump: starting
CrawlDb db: crawl/crawldb/
CrawlDb dump: done
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/JaiMataDiSports/apache-nutch-1.10$ bin/nutch readlinkdb crawl//linkdb/ -dump jlinkdb_output
LinkDb dump: starting at 2015-10-12 10:13:20
LinkDb dump: db: crawl//linkdb/
LinkDb dump: finished at 2015-10-12 10:13:22, elapsed: 00:00:01
```

Analysis -

As a result of the ReadDb command a file called part-00000 is created. For the education sites the size of the file was 1.1 MB for the Sports website the size was 902.6 KB.

When we crawl the output is in the crawl/crawldb directory. However, this output is not human readable. It is inside a directory called part-0000 which has data and index files.

But due to the ReadDb command we see human understandable output. This is easy for text processing. The file contains details about url, status, metadata and so on.

Excerpt from the part-0000 file -

```
http://nd.edu/    Version: 7
Status: 5 (db_redir_perm)
Fetch time: Wed Nov 11 03:08:40 EST 2015
Modified time: Wed Dec 31 19:00:00 EST 1969
Retries since fetch: 0
Retry interval: 2592000 seconds (30 days)
Score: 1.0
```

```
Signature: null
Metadata:
     Content-Type=text/html
   _pst_=moved(12), lastModified=0: http://www.nd.edu/
   _rs_=38

http://web.mit.edu/    Version: 7
Status: 2 (db_fetched)
Fetch time: Wed Nov 11 03:08:40 EST 2015
Modified time: Wed Dec 31 19:00:00 EST 1969
Retries since fetch: 0
Retry interval: 2592000 seconds (30 days)
Score: 1.0734382
Signature: 6359cabc4384a5706c108773b391e09b
Metadata:
     Content-Type=text/html
   _pst_=success(1), lastModified=0
   _rs_=45

http://web.mit.edu/aboutmit    Version: 7
Status: 5 (db_redir_perm)
Fetch time: Wed Nov 11 03:09:49 EST 2015
Modified time: Wed Dec 31 19:00:00 EST 1969
Retries since fetch: 0
Retry interval: 2592000 seconds (30 days)
Score: 0.039081886
Signature: null
Metadata:
     Content-Type=text/html
   _pst_=moved(12), lastModified=0: http://web.mit.edu/aboutmit/
   _rs_=101

http://web.mit.edu/aboutmit/    Version: 7
Status: 1 (db_unfetched)
Fetch time: Mon Oct 12 04:29:19 EDT 2015
Modified time: Wed Dec 31 19:00:00 EST 1969
Retries since fetch: 0
Retry interval: 2592000 seconds (30 days)
Score: 0.01135637
Signature: null
Metadata:
```

As a result of the invertlink command the links were inverted and linkDb was created. Now using the readlinkdb. Using the readlinkdb command an output file part-00000. This file contains information about inlinks. For education websites the size of this file is 2.1 MB and for Sports websites it was 2.8 MB.

This data is obtained from the crawl/linkdb directory which has current directory with an index and data file. Both are in non-human readable format. The commands convert this data to human understandable format which is easy for text processing.

An excerpt of the file for Sports Websites is.

```
http://bleacherreport.com/articles/feed    Inlinks:
 fromUrl: http://bleacherreport.com/nfl anchor:

http://www.fiba.com/    Inlinks:
 fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/CRO/ot/1/leagueProfile.html anchor: FIBA Logo
 fromUrl: http://www.fiba.com/media-centre/home anchor: FIBA.com
 fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/BRA/ot/1/leagueProfile.html anchor: FIBA Logo
 fromUrl: http://www.fiba.com/oceaniawomen/2015 anchor: FIBA.com
 fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/WNBA/ot/1/leagueProfile.html anchor: FIBA Logo
 fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/IRI/ot/1/leagueProfile.html anchor: FIBA Logo
 fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/LEBA/ot/1/leagueProfile.html anchor: FIBA Logo
 fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/BIH/ot/1/leagueProfile.html anchor: FIBA Logo
 fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/AUT/ot/1/leagueProfile.html anchor: FIBA Logo
 fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/WHUN/ot/1/leagueProfile.html anchor: FIBA Logo
```

```
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/BLR/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/EL/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/GER/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/WCZE/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/rankinggirls anchor: FIBA.com
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/INA/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/BEL/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/BALKAN/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/ARG/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/ISR2/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/WABA/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/ASEAN/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/world/u19women/2015 anchor: FIBA.com
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/ISL/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/TPE/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/EST/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/DEN/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/NBA/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/world/u19/2015 anchor: FIBA.com
fromUrl: http://www.fiba.com/rankingcombined anchor: FIBA.com
fromUrl: http://www.fiba.com/rankingmen anchor: FIBA.com
fromUrl: http://www.fiba.com/afrobasket/2015 anchor: FIBA.com
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/MEX/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/FRA/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/world/women/2014 anchor: FIBA.com
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/openNodeIDs/21192/selNodeID/21192/allResults.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/COPCC/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/afrobasketwomen/2015 anchor: FIBA.com
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/KAZ/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/GBR/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/PHI/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/oceania/2015 anchor: FIBA.com
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/QAT/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/UC/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/rankingboys anchor: FIBA.com
fromUrl: http://www.fiba.com/3x3U18WC/2015 anchor: FIBA.com
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/GER2/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/URU/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/ECM2/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/asiawomen/2015 anchor: FIBA.com
fromUrl: http://www.fiba.com/3x3WT/2015/abu-dhabi anchor: FIBA.com
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/GEO/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/asia/2015 anchor: FIBA.com
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/ADRIATIC/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/CAN/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/GRE/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/ELW/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/BUL/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/BALTIC_B/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/SUDAMERICA/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/hall-of-fame anchor: FIBA.com
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/CHN/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/WCRO/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/world/u17/2014 anchor: FIBA.com
fromUrl: http://www.fiba.com/world/u17women/2014 anchor: FIBA.com
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/HUN/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/ASIA_CHAMPIONS/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/ECW/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/news anchor: FIBA.com
fromUrl: http://www.fiba.com/rankingwomen anchor: FIBA.com
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/FIN/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/NBDL/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/VEN/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/GRE2/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/PUR/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/MEXLNPB/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/KOR/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/COL/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/events anchor: FIBA.com
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/WFRA/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/JPN/ot/1/leagueProfile.html anchor: FIBA Logo
fromUrl: http://www.fiba.com/3x3/federations-ranking anchor: FIBA.com
```

```
 fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/CZE/ot/1/leagueProfile.html anchor: FIBA Logo
 fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/BALTIC/ot/1/leagueProfile.html anchor: FIBA Logo
 fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/FRA2/ot/1/leagueProfile.html anchor: FIBA Logo
 fromUrl: http://www.fiba.com/pages/eng/fc/gameCent/p/leagueid/DOM/ot/1/leagueProfile.html anchor: FIBA Logo
 fromUrl: http://www.fiba.com/asia/u16women/2015 anchor: FIBA.com

http://www.fiba.com/3x3/federations-ranking    Inlinks:
 fromUrl: http://www.fiba.com/rankingwomen anchor: Federation Ranking
 fromUrl: http://www.fiba.com/events anchor: Federation Ranking
 fromUrl: http://www.fiba.com/asia/u16women/2015 anchor: Federation Ranking
 fromUrl: http://www.fiba.com/hall-of-fame anchor: Federation Ranking
 fromUrl: http://www.fiba.com/world/u17women/2014 anchor: Federation Ranking
 fromUrl: http://www.fiba.com/ anchor: Federation Ranking
 fromUrl: http://www.fiba.com/world/u17/2014 anchor: Federation Ranking
 fromUrl: http://www.fiba.com/news anchor: Federation Ranking
 fromUrl: http://www.fiba.com/3x3WT/2015/abu-dhabi anchor: Federation Ranking
 fromUrl: http://www.fiba.com/asiawomen/2015 anchor: Federation Ranking
 fromUrl: http://www.fiba.com/asia/2015 anchor: Federation Ranking
 fromUrl: http://www.fiba.com/oceania/2015 anchor: Federation Ranking
```

INLINK CALCULATION

The part-0000 text files outputted due to the readlinkdb command were analyzed first visually. After visual inspection a pattern was observed and then text processing was used to find the inlinks

For the Education Sites there were 24205 unlinks

```
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/apache-nutch-1.10/jlinkdb_output$ grep fromUrl: part-00000 | wc -l
24205
```

For the Sports Sites there were 33330 unlinks

```
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/JaiMataDiSports/apache-nutch-1.10/jlinkdb_output$ grep fromUrl: part-00000 | wc -l
33330
```

READSEG COMMAND and DUMPING OUTPUT of FETCHED DATA

The readseg   command dumps the content of a segment. We used it to dump the segment contents in a human readable format. It was visually analyzed.

This command takes as input the crawl/segments/<TimeStamp of the segment> as input and dumps a file which can be text processed

```
bin/nutch readseg -dump crawl/segments/20151012040927/ jreadseg_output
SegmentReader: dump segment: crawl/segments/20151012040927
SegmentReader: done
```

For the Education Websites it read dumped a file which was 29.3 MB in size and for the Sports Websites the file was 43.3 MB. These two files were later combined to obtain the dump.txt file of size 76.2 MB, which is used as input to the Second phase.

NOTE - These files were big in size and lot of text processing was required.
These have MetaData + HTML + ParsedData of the pages fetched.

OUTLINKS CALCULATION

It was seen that the readlinkdb tool could be used to find list of inlinks. To find the out-links we need to merge the segments and read the result.

The mergesegs command was used to merge the segments.

Then the merged segments were read using readseg command

The output file was visually inspected and text processed to find the number of outlinks.

It was found that there were 28700 out-links for the education website and 36263 for the sports one.

EDUCATION WEBSITE MERGE SEGMENTS

```
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/apache-nutch-1.10$ bin/nutch mergesegs jmergesegs crawl/segments/20151012040815/ crawl/segments/20151012040927/
Merging 2 segments to jmergesegs/20151012104533
SegmentMerger:   adding crawl/segments/20151012040815
SegmentMerger:   adding crawl/segments/20151012040927
SegmentMerger: using segment data from: content crawl_generate crawl_fetch crawl_parse parse_data parse_text
```

SPORT SITE MERGE SEGMENTS

```
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/JaiMataDiSports/apache-nutch-1.10$ bin/nutch mergesegs jmergesges crawl/segments/20151012042631/ crawl/segments/20151012042759/
Merging 2 segments to jmergesges/20151012105857
SegmentMerger:   adding crawl/segments/20151012042631
SegmentMerger:   adding crawl/segments/20151012042759
SegmentMerger: using segment data from: content crawl_generate crawl_fetch crawl_parse parse_data parse_text
```

SPORTS WEBSITE OUTLINKS

```
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/JaiMataDiSports/apache-nutch-1.10/joutlinks$ bin/nutch readseg -dump jmergesges/20151012105857/ joutlinks
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/JaiMataDiSports/apache-nutch-1.10/joutlinks$ grep Outlinks dump > joutlinkdetails.txt
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/JaiMataDiSports/apache-nutch-1.10/joutlinks$ awk '{s+=$0} END {print s}' joutlinkdetails.txt
36263
```

LINK DEPTHS ANALYSIS

EDUCATION WEB SITES

URLS with depth and their counts -

Depth of 8=   1

Depth of 7   9 - 1 = 8

Depth of 6   66 - 9 = 57

Depth of 5   277 - 66 = 211

Depth of 4   827 - 277 = 550

Depth of 3   1859 - 827 = 1032

Depth of 2   3523 - 1859 = 1664

Depth of 1   3787 - 3523 = 264

Total Number of URLS = 1 + 8 + 57 + 211 + 550 + 1032 + 1664 + 264 = 3787 which matches the initial statistics found using readdb.

This took as input the output of the readdb which was dumped into a text file.

Using regular expressions this was found the output and commands are below-

```
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/apache-nutch-1.10/joutput$ grep -o http://[a-zA-Z.0-9+-]*/ part-00000 | wc -l
3787
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/apache-nutch-1.10/joutput$ grep -o http://[a-zA-Z.0-9+-]*/[a-zA-Z0-9+-]*/
part-00000 | wc -l
3523
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/apache-nutch-1.10/joutput$ grep -o http://[a-zA-Z.0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-
Z0-9+-]*/ part-00000 | wc -l
1859
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/apache-nutch-1.10/joutput$ grep -o http://[a-zA-Z.0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-
Z0-9+-]*/[a-zA-Z0-9+-]*/ part-00000 | wc -l
827
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/apache-nutch-1.10/joutput$ grep -o http://[a-zA-Z.0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-
Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/ part-00000 | wc -l
277
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/apache-nutch-1.10/joutput$ grep -o http://[a-zA-Z.0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-
Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/ part-00000 | wc -l
66
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/apache-nutch-1.10/joutput$ grep -o http://[a-zA-Z.0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-
Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/ part-00000 | wc -l
9
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/apache-nutch-1.10/joutput$ grep -o http://[a-zA-Z.0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-
Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/ part-00000 | wc -l
1
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/apache-nutch-1.10/joutput$ grep -o http://[a-zA-Z.0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-
Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/ part-00000 | wc -l
0
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/apache-nutch-1.10/joutput$
```

SPORTS WEBSITE ANALYSIS

URLS with depths and counts -

Depth of 11= 6

Depth of 10 81 - 6 = 75

Depth of 9 97 - 81 = 16

Depth of 8 102 - 97 = 5

Depth of 7 111 - 102 = 9

Depth of 6 111 - 111 = 0

Depth of 5 193 - 111 = 82

Depth of 4 492 - 193 = 299

Depth of 3 1491 - 492 = 999

Depth of 2 2646 - 1491 = 1155

Depth of 1 2883 - 2646 = 237

Total Number of URLS = 6 + 75 + 16 + 5 + 9 + 0 + 82 + 299 + 999 + 1155 + 237 = 2883 which matches the initial statistics found using read

NOTE - There less urls at greater value of depth and more on top. This is obvious because nutch uses frontier expansion of Breadth First Search by default.

```
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/JaiMataDiSports/apache-nutch-1.10/joutput$ grep http:// part-00000 | wc -l
2883
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/JaiMataDiSports/apache-nutch-1.10/joutput$ clear

jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/JaiMataDiSports/apache-nutch-1.10/joutput$ grep -o http://[a-zA-Z.0-9+-]*/ part-00000 | wc -l
2883
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/JaiMataDiSports/apache-nutch-1.10/joutput$ grep -o http://[a-zA-Z.0-9+-]*/[a-zA-Z0-9+-]*/ part-00000 | wc -l
2646
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/JaiMataDiSports/apache-nutch-1.10/joutput$ grep -o http://[a-zA-Z.0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/ part-00000 | wc -l
1491
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/JaiMataDiSports/apache-nutch-1.10/joutput$ grep -o http://[a-zA-Z.0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/ part-00000 | wc -l
492
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/JaiMataDiSports/apache-nutch-1.10/joutput$ grep -o http://[a-zA-Z.0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/ part-00000 | wc -l
193
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/JaiMataDiSports/apache-nutch-1.10/joutput$ grep -o http://[a-zA-Z.0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/ part-00000 | wc -l
111
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/JaiMataDiSports/apache-nutch-1.10/joutput$ grep -o http://[a-zA-Z.0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/ part-00000 | wc -l
111
```

```
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/JaiMataDiSports/apache-nutch-1.10/joutput$ grep -o http://[a-zA-Z.0-9+-]*/[a-
zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/ part-00000 | wc -l
102
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/JaiMataDiSports/apache-nutch-1.10/joutput$ grep -o http://[a-zA-Z.0-9+-]*/[a-
zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/ part-00000 |
wc -l
97
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/JaiMataDiSports/apache-nutch-1.10/joutput$ grep -o http://[a-zA-Z.0-9+-]*/[a-
zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/
part-00000 | wc -l
81
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/JaiMataDiSports/apache-nutch-1.10/joutput$ grep -o http://[a-zA-Z.0-9+-]*/[a-
zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/
[a-zA-Z0-9+-]*/ part-00000 | wc -l
6
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/JaiMataDiSports/apache-nutch-1.10/joutput$ grep -o http://[a-zA-Z.0-9+-]*/[a-
zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/
[a-zA-Z0-9+-]*/[a-zA-Z0-9+-]*/ part-00000 | wc -l
0
```

WORD COUNT -

We visually inspected the read segment’s output. Then used text processing to extract the parsed text. We then counted the frequency of words in the text.

The example with explanation of command used for this purpose-

The command:
grep -o -E '\w+’ part-00000

The explanation:
grep:
-o only prints the portion of the line that matches that pattern.
-E searches for a certain pattern.
'\w+’ indicates that “word” is the pattern to be matched.

Education Websites

```
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/apache-nutch-1.10/jreadseg_output$ grep -A 1 ParseText:: dump > jParsedText.txt
```

Remove ParseText:: using text editor

```
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/apache-nutch-1.10/jreadseg_output$ grep -o -E '\w+' jParsedText.txt | sort -f |
uniq -c | sort -bnr > jWordCount.txt
```

Output - (ONLY SOME PART is HERE)

```
 10762 and
  9831 the
  9408 of
  5952 to
  4346 in
  3876 for
  3674 a
  3462 Caltech
  3183 University
  2423 s
  2228 Research
```

2005	School
1991	at
1820	on
1818	is
1719	2015
1717	The
1689	Oct
1624	Center
1475	with
1418	Campus
1334	News
1319	06
1230	Rutgers
1219	Students
1208	ly
1198	that
1180	Graduate
1126	students
1123	Penn
1117	ow
1089	Admissions
1058	Faculty
1022	Education
995	research
993	Resources
992	Flickr
992	12
988	t
963	by
959	Health
955	Academic
929	A
920	Events
905	Harvard
898	co
895	Rice
876	Alumni
873	Services
872	are
867	Undergraduate
864	Programs
855	or
854	http
837	Staff
836	from
828	About
822	New
820	as
819	Calendar
802	Sciences
755	Student
755	Our
753	Arts
748	Studies
743	Engineering
740	Office
732	more
725	our
716	Visit
701	Academics
699	Public
697	Aid
680	Search
680	Contact
658	Vanderbilt
655	an
649	you
629	Information
619	Media
618	Institute
600	Science
591	campus

587	Columbia
583	Athletics
570	Yale
570	College
567	Northwestern
567	be
563	Life
559	Directory
558	Give
533	More
533	faculty
530	will
527	their
526	Giving
523	History
522	Financial
516	about
513	Brown
511	new
507	this
502	Z
501	Maps
491	International
490	Us
488	Program
485	programs
485	Global
483	Facebook
478	Twitter
473	Social
470	Washington
462	Learning
448	world
448	edu
446	Offices
446	Libraries
445	Online
444	your
438	322
430	Library
420	Medicine
416	information
410	President
409	Community
408	Continuing
402	Leadership
400	Safety
400	content
399	Policy
399	Parents
391	community
387	has
384	all
382	Schools
374	We
363	we
363	Professional
361	Medical
359	Copyright
354	main
348	S
347	have
347	01

The example with explanation of the command used for this purpose-

The command:

```
/Downloads/apache-nutch-1.10/runtime/local/out2db$ grep -o -E '\w+' part-00000 | sort -f | uniq -c | sort -bnr
```

The explanation:

grep:

-o only prints the portion of the line that matches that pattern.

-E searches for a certain pattern.

'\w+' indicates that "word" is the pattern to be matched.

sort :

-f tells sort to ignore case when comparing words.

-bnr sorts in a numeric reverse order, while ignoring whitespaces.

Uniq:

-c counts the occurrences.

SPORTS WEBSITES FREQUENCY OF WORDS -

```
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/JaiMataDiSports/apache-nutch-1.10/jreadseg_output$ grep -A 1 ParseText:: dump > jParsedText.txt
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/JaiMataDiSports/apache-nutch-1.10/jreadseg_output$ vi jParsedText.txt

[3]+  Stopped                 vi jParsedText.txt
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/JaiMataDiSports/apache-nutch-1.10/jreadseg_output$ grep -o -E '\w+' jParsedText.txt | sort -f | uniq -c | sort -bnr > jWordCount.txt
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/JaiMataDiSports/apache-nutch-1.10/jreadseg_output$
```

OUTPUT (ONLY SOME PART IS HERE)

```
4863 the
   3417 U
   3148 of
   3053 0
   2637 1
   2427 s
   2343 and
   2141 to
   2110 Stories
   2085 About
   2039 n
   2018 Roster
   1922 2015
   1878 Matches
   1852 in
   1807 2
   1764 Stats
   1764 Latest
   1716 Tournaments
   1625 a
```

```
1508 Cup
1437 Coaching
1417 Women
1377 t
1359 The
1292 Soccer
1292 News
1270 5
1259 3
1231 World
1222 Staff
1218 i
1194 Tickets
1123 4
1033 for
1028 MNT
1027 League
1018 S
1009 FIFA
1006 NBLC
 990 u
 983 r
 973 on
 948 WNT
 901 All
 897 2014
 869 15
 848 UEFA
 844 with
 839 6
 839 10
 838 Home
 838 f
 768 20
 757 Football
 744 Club
 743 FIBA
 719 17
 699 this
 695 at
 684 function
 655 Game
 654 return
 643 8
 629 16
 625 NBCSN
 622 19
 620 com
 620 18
 616 Canada
 595 ET
 591 A
 590 vs
 586 Men
 573 e
 568 National
 568 Islands
 567 Republic
 560 23
 558 NFL
 553 Live
 539 Team
 539 Basketball
 533 o
 532 7
 531 11
 525 is
 518 New
 505 USA
 504 Search
 504 NBL
 501 2013
```

```
498 Real
493 9
487 by
485 Week
480 Standings
468 Store
465 Season
461 FOX
453 Liga
444 Archive
438 Teams
429 from
427 Championship
425 12
424 your
422 Members
419 Bernabéu
419 be
415 Schedule
414 14
411 Video
411 or
409 Madrid
396 Contact
```

HIGHEST FREQUENCY URL

We use similar technique as used for word frequency to find the highest frequency urls.The higher the frequency the more popular and the more connected.We take as input the readdb command's output's parse-0000 dump.

The example with explanation of the command used for this purpose-

The command combination:

```
grep -o -E "\<"http:/".*\>/" part-00000 | sort -f | uniq -c | sort -bnr
```

The explanation:

"\<"http:/".*\>/" will consider only the URLs as a pattern to be parsed.

EDUCATION CENTER

URL RANKING

```
jaimatadi@jaimatadi-laptop:~/Documents/JaiMataDi/apache-nutch-1.10/joutput$ grep -o -E "\<"http:/".*\>/" part-00000 | sort -f | uniq
-c | sort -bnr > jLinkRank.txt
```

OUTPUT (SOME PART ONLY)

```
126 http://www.caltech.edu/content/
    79 http://www.washington.edu/students/crscat/
    76 http://www.caltech.edu/node/
    68 http://www.caltech.edu/news/
    53 http://www.vanderbilt.edu/
    46 http://www.upenn.edu/node/
    40 http://www.caltech.edu/file/
    39 http://www.upenn.edu/spotlights/
    38 http://www.columbia.edu/content/
```

```
36 http://www.rutgers.edu/about/
33 http://www.rutgers.edu/node/
32 http://www.caltech.edu/news/tag_ids/
32 http://web.mit.edu/facts/
31 http://www.rutgers.edu/academics/
31 http://www.columbia.edu/node/
31 http://web.mit.edu/research/topic/
28 http://www.harvard.edu/
25 http://www.yale.edu/about/
25 http://www.washington.edu/students/reg/
24 http://www.vanderbilt.edu/atoz/letter/
24 http://www.upenn.edu/penna-z/
22 http://www.northwestern.edu/studyabroad/programs/europe/
21 http://www.vanderbilt.edu/atoz/tag/
21 http://www.upenn.edu/
19 http://www.columbia.edu/cu/opir/abstract/
19 http://web.mit.edu/
18 http://www.vanderbilt.edu/career/tools/
18 http://www.rice.edu/
16 http://www.vanderbilt.edu/career/
16 http://www.upenn.edu/pennnews/
16 http://web.mit.edu/community/topic/
15 http://www.washington.edu/research/.SITEPARTS/.documents/.or/
15 http://www.brown.edu/academics/engineering/about/
15 http://www.brown.edu/
14 http://www.upenn.edu/about/
14 http://www.caltech.edu/
14 http://web.mit.edu/industry/
13 http://www.upenn.edu/programs/
13 http://www.upenn.edu/pennnews/current/
12 http://www.yale.edu/hronline/benefits/
12 http://www.yale.edu/
12 http://www.vanderbilt.edu/financialaid/
11 http://www.rice.edu/unconventional/
11 http://www.brown.edu/academics/engineering/
11 http://web.mit.edu/staff/
10 http://www.yale.edu/gateways/
10 http://www.upenn.edu/pennnews/current/calendar/%40--2015-11-11/
10 http://www.upenn.edu/life-at-penn/
10 http://www.northwestern.edu/about/our-people/
10 http://www.harvard.edu/about-harvard/
10 http://www.columbia.edu/files/columbia/content/
10 http://www.brown.edu/about/
10 http://web.mit.edu/staff/business/
 9 http://www.yale.edu/hronline/careers/
 9 http://www.upenn.edu/sites/default/files/research-at-penn/pdf/
 9 http://www.stanford.edu/
 9 http://www.northwestern.edu/studyabroad/programs/americas/
 9 http://www.brown.edu/campus-life/events/family-weekend/
 9 http://www.brown.edu/campus-life/
 9 http://www.brown.edu/admission/undergraduate/
 9 http://www.brown.edu/academics/
 8 http://www.vanderbilt.edu/work-at-vanderbilt/
 8 http://www.upenn.edu/pennnews/experts/
 8 http://www.upenn.edu/computing/security/advisories/
 8 http://www.northwestern.edu/studyabroad/programs/asia/
 8 http://www.northwestern.edu/globalhealthstudies/about/
 8 http://www.harvard.edu/media-relations/
 8 http://www.brown.edu/academics/engineering/about/events/2015-10/
 7 http://www.vanderbilt.edu/financialaid/undergraduate/
 7 http://www.upenn.edu/services/
 7 http://www.upenn.edu/president/meet-president/
 7 http://www.upenn.edu/computing/security/
 7 http://www.stanford.edu/academics/
 7 http://www.northwestern.edu/studyabroad/programs/africa/
 7 http://www.harvard.edu/on-campus/commencement/
 7 http://www.harvard.edu/about-harvard/harvard-glance/
 7 http://web.mit.edu/mitpsc/whoweare/
 6 http://www.yale.edu/medicine/
 6 http://www.washington.edu/students/gencat/front/
 6 http://www.washington.edu/
 6 http://www.upenn.edu/sustainability/sustainability-themes/
```

```
      6 http://www.uchicago.edu/community/
      6 http://www.northwestern.edu/newscenter/stories/2015/10/
      6 http://www.northwestern.edu/globalhealthstudies/faculty/
      6 http://www.northwestern.edu/diversity/
      6 http://www.harvard.edu/visitors/
      6 http://www.brown.edu/gateway/
      6 http://www.brown.edu/academics/professional/
      6 http://www.brown.edu/academics/gradschool/
      6 http://www.brown.edu/about/administration/global-engagement/
      6 http://wustl.edu/academics/
      6 http://web.mit.edu/staff/culture/
      5 http://www.yale.edu/hronline/
      5 http://www.vanderbilt.edu/ResEd/main/
      5 http://www.vanderbilt.edu/nashville/
      5 http://www.vanderbilt.edu/isss/wp-content/uploads/
      5 http://www.vanderbilt.edu/international/funding/
      5 http://www.vanderbilt.edu/financialaid/loans-payment/
      5 http://www.upenn.edu/president/about-presidency/
      5 http://www.upenn.edu/pennnews/current/2015-10-08/latest-news/
      5 http://www.uchicago.edu/students/
      5 http://www.stanford.edu/about/
      5 http://www.princeton.edu/main/
      5 http://www.northwestern.edu/studyabroad/programs/middle-east/
      5 http://www.northwestern.edu/academics/
      5 http://www.northwestern.edu/about/
      5 http://www.northwestern.edu/
      5 http://www.harvard.edu/on-campus/visit-harvard/
      5 http://www.harvard.edu/on-campus/
```

SPORTS CENTER RANKINGs

We use similar commands as used for education center.

OUTPUT (SOME PART ONLY)

```
186 http://www.foxsports.com/college-football/
    113 http://www.foxsports.com/soccer/
     97 http://www.foxsports.com/mlb/
     93 http://www.foxsports.com/nfl/
     93 http://www.foxsports.com/golf/
     66 http://www.isaa.org/popups/vendors/
     65 http://www.foxsports.com/nascar/
     62 http://www.nblcanada.com/nblcanada/images/
     45 http://www.foxsports.com/nba/
     39 http://www.foxsports.com/golf/story/
     36 http://www.foxsports.com/soccer/story/
     34 http://www.foxsports.com/college-football/story/
     31 http://www.foxsports.com/nascar/photos/
     30 http://www.foxsports.com/
     29 http://www.isaa.org/popups/
     26 http://www.nbcsports.com/
     24 http://www.isaa.org/
     23 http://www.nbcsports.com/video/
     22 http://www.nfl.com/scores/2015/
     22 http://www.foxsports.com/college-football/photos/
     20 http://www.nfl.com/
     20 http://www.foxsports.com/soccer/lists/
     19 http://www.realmadrid.com/
     19 http://www.nbcsports.com/node/
     19 http://www.fiba.com/File/
     18 http://www.ussoccer.com/
     18 http://www.foxsports.com/nfl/story/
     16 http://www.foxsports.com/mlb/story/
     14 http://www.foxsports.com/tag/
     14 http://www.foxsports.com/nascar/shake-and-bake/
     14 http://www.foxsports.com/motor/story/
     12 http://www.foxsports.com/soccer/paper-chase/
```

```
12 http://www.foxsports.com/nfl/photos/
12 http://www.foxsports.com/nba/story/
12 http://www.foxsports.com/nascar/story/
12 http://www.fiba.com/
11 http://www.ussoccer.com/womens-national-team/
11 http://www.foxsports.com/college-football/outkick-the-coverage/
10 http://www.ussoccer.com/mens-national-team/
 8 http://www.ussoccer.com/us-under19-mens-national-team/
 8 http://www.realmadrid.com/zh/football/
 8 http://www.realmadrid.com/zh/
 8 http://www.realmadrid.com/ja/football/
 8 http://www.realmadrid.com/id/
 8 http://www.realmadrid.com/fr/
 8 http://www.realmadrid.com/en/football/
 8 http://www.realmadrid.com/en/
 8 http://www.realmadrid.com/ar/football/
 8 http://www.realmadrid.com/ar/
 8 http://www.nbcsports.com/video/league/
 8 http://www.fifa.com/ballon-dor/organisation/
 7 http://www.wimbledon.com/en_GB/atoz/
 7 http://www.ussoccer.com/us-under23-mens-national-team/
 7 http://www.ussoccer.com/us-under20-womens-national-team/
 7 http://www.ussoccer.com/us-under20-mens-national-team/
 7 http://www.ussoccer.com/us-under19-womens-national-team/
 7 http://www.ussoccer.com/us-under18-mens-national-team/
 7 http://www.realmadrid.com/sobre-el-real-madrid/historia/
 7 http://www.realmadrid.com/pt/sobre-o-real-madrid/historia/
 7 http://www.realmadrid.com/pt/futebol/jogos/
 7 http://www.realmadrid.com/pt/
 7 http://www.realmadrid.com/ja/
 7 http://www.realmadrid.com/id/sepak-bola/pertandingan/
 7 http://www.realmadrid.com/futbol/partidos/
 7 http://www.realmadrid.com/fr/football/matchs/
 7 http://www.realmadrid.com/fr/a-propos-du-real-madrid/histoire/
 7 http://www.nbcsports.com/live-extra/
 7 http://www.isaa.org/pdfs/
 7 http://www.foxsports.com/nfl/video/
 7 http://www.foxsports.com/nascar/video/
 7 http://www.foxsports.com/golf/usga/story/
 7 http://www.fifa.com/
 6 http://www.wimbledon.com/en_GB/wimbledonfoundation/
 6 http://www.ussoccer.com/us-under23-womens-national-team/
 6 http://www.ussoccer.com/us-under18-womens-national-team/
 6 http://www.realmadrid.com/zh/about-real-madrid/history/
 6 http://www.realmadrid.com/ja/about-real-madrid/history/
 6 http://www.realmadrid.com/id/basket/pertandingan/
 6 http://www.realmadrid.com/fr/zone-vip/
 6 http://www.realmadrid.com/en/vip-area/
 6 http://www.realmadrid.com/en/basketball/
 6 http://www.realmadrid.com/en/about-real-madrid/history/
 6 http://www.realmadrid.com/ar/about-real-madrid/history/
 6 http://www.foxsports.com/golf/usga/
 6 http://www.foxsports.com/college-football/lists/
 6 http://www.fifa.com/womens-football/
 5 http://www.realmadrid.com/zh/vip-area/
 5 http://www.realmadrid.com/zh/basketball/games/
 5 http://www.realmadrid.com/zh/basketball/
 5 http://www.realmadrid.com/sobre-el-real-madrid/el-club/
 5 http://www.realmadrid.com/pt/futebol/
 5 http://www.realmadrid.com/pt/basquetebol/jogos/
 5 http://www.realmadrid.com/pt/basquetebol/
 5 http://www.realmadrid.com/pt/area-vip/
 5 http://www.realmadrid.com/ja/vip-area/
 5 http://www.realmadrid.com/ja/basketball/games/
 5 http://www.realmadrid.com/ja/basketball/
 5 http://www.realmadrid.com/id/tentang-real-madrid/sejarah/
 5 http://www.realmadrid.com/id/sepak-bola/
 5 http://www.realmadrid.com/id/keramahtamahan/
 5 http://www.realmadrid.com/futbol/
```

Phase 2 - Search Engine and Indexer

The purpose of second phase was to basically build a search engine and indexer which accepted the input of Phase 1 and and provided an Inverted Index and searching and ranking features. Since nutch fetches webpages, therefore the motive behind building an inverted index was to enable search.

An Inverted Index is a key-value index where the key is a word and the value is the document (number) which contains that word, in a corpus (collection of documents). This enables searching - where a user can provide a search term consisting of one or more words which are then fed as input to the search engine and by utilizing the inverted index, the corresponding documents (numbers) are retrieved.

However, the task of the search engine doesn't end here, this is because a word (e.g. the) could potentially exists in too many documents. Therefore, providing an appropriate ordering by using a Ranking Algorithm is required. The objective of the Ranking Algorithm is that, once using the inverted index the documents (numbers) which contain a word have been identified, it ranks the documents (URLs associated with documents) from the most relevant to the least relevant.

Even though the relevance of a document given a search term is a topic, which is beyond the scope of this document, however, it is greatly significant. Without a good ranking algorithm a search engine is, at best, a giant index which gives the user all the documents which contain word(s) entered by the user. One of the key factors which affects the success of a search engine is the effectiveness with which its ranking algorithm can capture the relevance of a document given a search term and rank (order) the documents (URLs associated with the documents).

2.1 Development Environment of the Search Engine and Indexer

The Development Environment for the Search Engine and Indexer consisted of -

1. Programming Language - *Java* SDK 8 (the binaries were compiled to be compatible with JDK 6 because of the JDK 6 available in the Deployment Environment). Java was considered as a natural choice because of the *Write-Once-Run-Anywhere* nature of the Java platform and was further supported by the fact the crawler - nutch is also written in Java (considering future possibility of integrating the two).

2. Integrated Development Environment used - *Eclipse Mars*.

2.2 File Processing

The input from Phase 1 i.e. dump.txt had more than 1.6 million lines of text consisting of all the data, parsed data, urls and metadata and much more information fetched by the crawler. The objective of File Processing was to extract the parsed text of the web pages and the corresponding web pages. So that once we had each web page (with its URL) and the parsed text associated with it, we could start building an inverted-index on it.

It was a difficult task to process files, because there was lot of noise in the form of URLs, metadata , timestamps and information which did not contribute to our goal of extracting web pages' parsed texts and respective URLs.

The dump.txt was first visually inspected and based on visual inspection certain patterns could be observed. The next step was to programmatically implement what had been the observed during visual inspections to extract the parsed web pages' contents and associated URLs

In File Processing -

1. FileMetadataGenerator.java - accepts any file (obtained from nutch's readseg command i.e. the dump.txt) and generates a metadata which used in the file processing algorithm.

2. FileCleaner.java - It accepts the metadata generated by FileMetadataGenerator.java and the original file and cleans it as per algorithm.

3. FileSplitter - This file accepts the outputs after *two* passes of metadata generation and cleaning and then splits the result to generate the documents - each of which corresponds to **one web page and has all that page's fetched parsed text**. All the output files are text files.

The output obtained were 687 files. Each of the files corresponds to a specific web-page. A file consists of the URL (of the webpage) and the parsed text contained in the web page.

Note - all the files in the existing implementation were obtained from the crawling and fetching performed in the month of October 2015. The dump.txt used as input belongs to the October 2015. Since then, the content of several web pages has changed. Therefore this needs to be factored in, when looking at results.

Despite using an effective algorithm for file processing, the visual inspection of all 687 parsed text file revealed that 6 had either no data in it or junk values - this makes the algorithm more than 99% accurate in separating useful data from junk. For the rest the data was perfect as expected.

However, when the above 6 links were visited either they no longer existed or were file links. One of the links had the same data, which we thought of as “junk data” during visual inspection, hence this couldn’t be attributed as deficiency of the algorithm. Rather this should be viewed as proof of effectiveness of the algorithm. If the web page contains junk, which is then parsed by the nutch, the indexer’s job is to merely index the page, it is not designed for linguistic filtering or processing, which is beyond the scope of this course.

Time Consumed - It takes around 1500 ms

OUTPUT - 1 dump.txt (76.2MB) ——-> 687 text documents (4.2MB) (at /<directory-root>/files/)

2.3 Token Generation

In lexical analysis, tokenization is the process of breaking a stream of text, up into words, phrases, symbols, or other meaningful elements called tokens. The list of tokens becomes input for further processing such as parsing or text mining.

Since the objective is to create an Inverted Index, we therefore tokenize to efficiently create an index. The purpose of the Inverted Index is to provide a mapping between a word (key) and the document (value(s) - the document number of the document which contains the word) from the corpus (the collection of documents). Tokenization helps in efficient indexing.

For this stage we take the files obtained from the previous stage (File Processing) as input and then for each file -

1. Read the file

2. Create a Set of words in the file. This is because a Set has the mathematical property of having only unique elements, i.e. a Set can not contain an element more than once. A set can not have redundancy by virtue of its mathematical properties. This property of not having redundancy is utilized for fast index creation.

3. Create a new file containing tokens of the input file. These tokens are then used as keys of the Inverted Index.

The reader by think -

Why bother tokenizing ?

Why not use the files directly to form the Inverted-Index ?

Isn't tokenizing an overhead ?

Tokenizing helps to efficiently index because during indexing the token files can be used and the token files efficiently answer a fundamental question -

Que - Does this particular document contain a word (search-key) ?
Ans - Yes/No ?

Which is all that is required to create an Inverted Index. The fact that the tokenized files - files containing only the tokens for a corresponding file obtained from the File Processing stage, do not have redundancy make Indexing faster and efficient. The tokenized files also serve as inputs to the next stage where we describe Stemming.

However, a limitation of the tokenized files is that they can just answer the question about the existence of a word in a document; they *can not reveal the number of occurrences of a word inside a document*. Since the number of occurrences of a word in a document, is an input in many ranking algorithms (e.g. TFIDF , and Document Quality - both of which are used in this search engine), the tokenized files are of no use for such algorithms.

Note - In the first version of the Search Engine - the tokenized files were used during Indexing. However, in the existing version they just act as input for Stemming (described in the next section). In the current version Indexing is combined with tokenization, this helped to reduce the Indexing time from 90s to 10-15s. (6X to 9X times faster).

OUTPUT - Around 600 ms 687 tokenized files 2.9 MB at <directory-root>/tokenj/

2.4 Stemming

Stemming tries to reduce a word to its roots. The purpose is that, it helps to broaden the scope of search in a situation where the word(s), provided as input by the user, are not "as it is" present in the corpus. e.g. biological and biology both have the same lexical root. Thus it could potentially help to locate a word , which has the same root as the word(s) the user is looking for, but is not present "as it is" in the corpus.

Stemmer.java is the class responsible for this. This class was obtained from http://tartarus.org/martin/PorterStemmer/java.txt

The Stemmer.java was appropriately modified to fit our needs. The methods - stemTokens() and createDifferentStemmedTokens() inside the SearchEngine.java class are responsible for stemming. For each tokenFile that is there a stemmedtokenFile is produced. While producing a stemmedtokenFile it is ensured that, if the stemmed token is same as the original token, then it is *not* copied to the stemmedtokenFile. The reason is that, copying the same word only leads to *redundancy* - which *degrades performance* during search through files. During the search when the index does not contain the word being looked for, then these stemmedtokenfiles are used.

Since the stemmed file will contain only those stemmed words which are different than their original non-stemmed versions, some stemmedtokenFiles can be empty. This implies that all words of the original source tokenFile after stemming remained the same and hence none of the stemmed words were copied to the stemmedtokenfile.

The reader may think -

What is the point having an empty stemmedtokenfile ?

Should not such files be discarded ?

The point is that it is not just a file - it acts a metadata. When we look at a parsed web document in the e.g. *<directory-root>/filesj/file.txt* the application guarantees that there would be a corresponding tokenFile.txt and stemmedtokenFile.txt - these two act as metadata for the original parsed web document. This helps during searches and index creation. This could have future applications in further research.

OUTPUT approximately processed in 300 ms at <directory-root>/differenstemmedtokenj/

Note- the directory is named as “differentstemmedtoken” - which conveys that in this directory only those tokens are there which after stemming differ from the original token given as input to the process of stemming , as already explained.

2.5 Index Generation

The Index Generation is one of the most critical objectives to be fulfilled. We create an Inverted-Index which is a key-value mapping from words (keys) to document numbers (value). The document number would be then used to find the document, in the corpus, and the document would contain the URL, to retrieve which is the ultimate goal of the Indexing.

First the generateIndices() method of the IndexGenerator class is called. This method calls the getSortedWordCounter() method of the same class and the getSortedWordCounter() method scans through all parsed file in the <directory-root>/filesj/ - in this case 687 files to generate a HASHMAP<STRING,INTEGER> which maps every distinct string to its number of occurrences across all parsed 687 pages. Then the application sorts them from Least Word Count to High Word Count - ascending order by word count - this hashmap - sortedWordCounter is used to calculate - lowFrequencyWords and highFrequencyWords and total words.

During Indexing we maintain two sets lowFrequencyWords and highFrequencyWords. These are used for Document Quality algorithm that we use during ranking.

1. lowFrequencyWords - These are words which occur just 1% in the entire corpus.

2. highFrequencyWords - The remaining words which constitute the remaining 99% of the corpus.

Filtration - This increases the efficiency of the indexing . For this, a primitive and augmented filter is used in the same class. The primitive filter consists of digits and alphabets - it is then transformed into an augmented filter by adding all 255 ascii values to it and numbers. Locations are - Primitive Filter in LINE 29 - IndexGenerator.java and Augmented Filter at LINE 130 and LINE 133 - IndexGenerator.java. If filtration is not used, then garbage values occupy the index - these values *do not* contribute to the search process.

Statistics about words

1. **Total** number of words in the sortedWordCounter = **411598**

2. **High** Frequency words **10780**

3. **Low** Frequency words **6584**

4. Number of **distinct** words = **17364** e.g. the word the occurs more than 10000 times

Finally by utilizing the methods of the class IndexGenerator the Inverted-Index is generated and is called “primaryIndex.txt” and is stored at <directory-root>/primaryindexj/

STATISTICS about primaryindex.txt

1. *TYPE - KEY,VALUE MAPPINGS key - the word , value - the document number where the word occurs*

2. *SIZE - 1.1 MB (NOTE the size of all 687 parsed web pages combined was 4.2 MB) => InvertedIndex here occupies a quarter of the space as all the parsed data.*

3. *LINES - 34728 which is correct because it is 2 times the number of distinct words in the parsed files - which is 17364. One line to store the word as KEY and another line to store all the document numbers where the word (key) occurs as VALUEs.*

It is vital for the normal operations of the application that, every time the application runs , it first checks if this file is in its correct location - <directory-root>/primaryindexj/primaryindex.txt If the application does not find the file at its designated location, then it is assumed by the application that it is the first time the application is being run and that indexing never ever took place. Therefore all processes- FILE PROCESSING, TOKEN GENERATION, STEMMING, INDEX GENERATION , SCORING are performed from scratch. If in the beginning , the application detects this file is present at its designated location - it assumes that this file was created during some previous run and the application directly loads the content of the Inverted Index into the main memory for usage during answering search queries.

CONCLUSION - No primaryIndex.txt => File Processing, Token Generation,
Stemming, Index Generation,
Scoring done from scratch.

In the First Version of the Search Engine and Indexer - Indexing used to take 10 minutes
In the Second Version - Indexing used to take about 90 seconds
In the Current Version - Indexing takes just 10 - 15 seconds

We have been able to speed up the creation of Inverted-Index by about 40 X to 60 X over the previous version.

2.6 Scoring

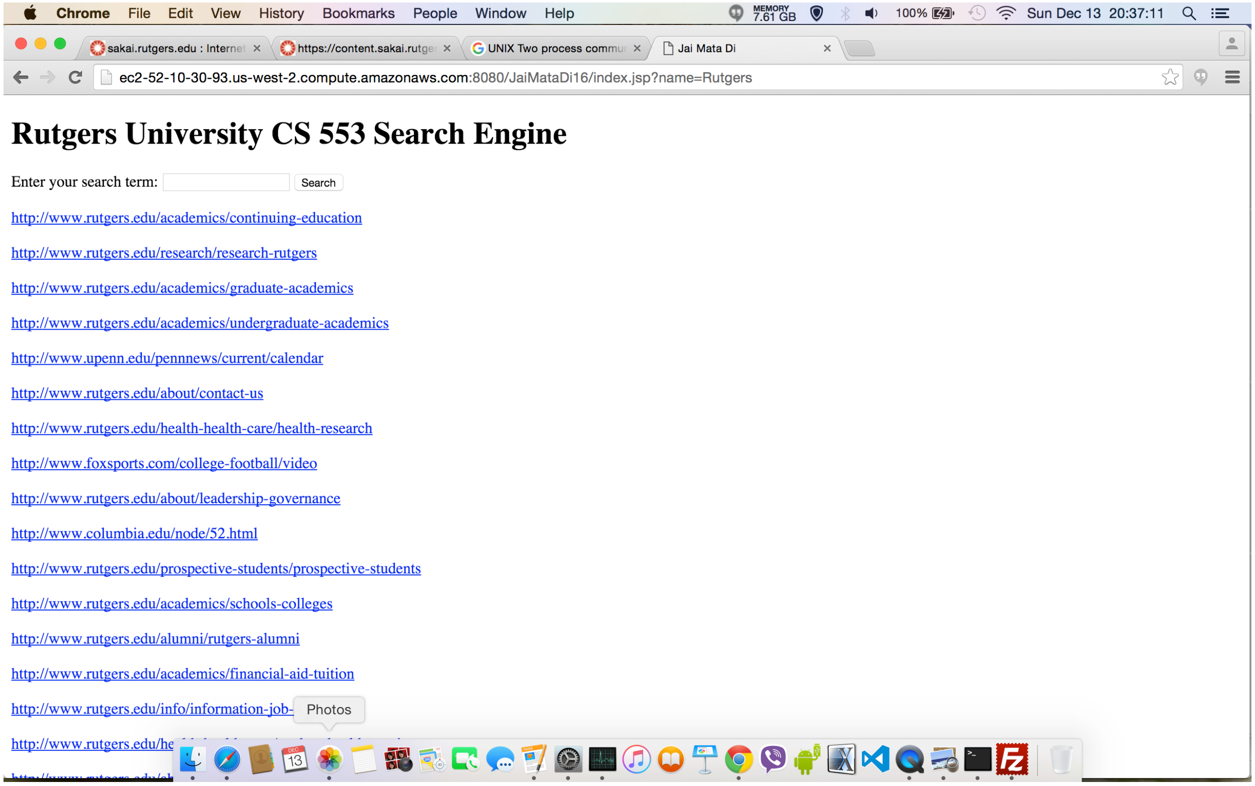
The Scoring is responsible for pre-processing of the documents in corpus. We use a score - Document Quality during ranking. The notion of Document Quality is -

Words which occur less frequently convey more semantic weight (*more notions packed inside a single word*) than frequently occurring words such as “of” , “for”, “the” etc. Therefore we say that words which constitute close to 1% of the total number of words in the entire corpus are lowFrequencyWords (See 2.5 for details) and the rest are called highFrequencyWords.

Document Quality of a document = lowFrequencyWords / (total number of word in a document)

The higher the ratio (on the R.H.S) the higher the Document Quality. DQ constitutes 60% of weightage during assigning total score to a page/document. Since the DQ of a document is independent of a search term, therefore it remains unchanged throughout the lifetime of a document in the corpus. Therefore, we pre-process DQ for all the parsed web pages stored in the *filesj* directory when the application runs for the first time. In subsequent usages these scores are directly loaded in to the main memory. This saves precious CPU cycles. The reader would have noticed that, this technique is similar to the one that is used for Inverted Index, where once the index is created, during the first run of the application, the Inverted Index is thereafter loaded directly into the main memory during subsequent usages.

Phase 3 - Web Interface


Chrome File Edit View History Bookmarks People Window Help
MEMORY 7.61 GB
100%
Sun Dec 13 20:37:11
sakai.rutgers.edu : Internet
https://content.sakai.rutge
UNIX Two process commu
Jai Mata Di
ec2-52-10-30-93.us-west-2.compute.amazonaws.com:8080/JaiMataDi16/index.jsp?name=Rutgers
Photos

Rutgers University CS 553 Search Engine

Enter your search term: [ ] Search

http://www.rutgers.edu/academics/continuing-education

http://www.rutgers.edu/research/research-rutgers

http://www.rutgers.edu/academics/graduate-academics

http://www.rutgers.edu/academics/undergraduate-academics

http://www.upenn.edu/pennnews/current/calendar

http://www.rutgers.edu/about/contact-us

http://www.rutgers.edu/health-health-care/health-research

http://www.foxsports.com/college-football/video

http://www.rutgers.edu/about/leadership-governance

http://www.columbia.edu/node/52.html

http://www.rutgers.edu/prospective-students/prospective-students

http://www.rutgers.edu/academics/schools-colleges

http://www.rutgers.edu/alumni/rutgers-alumni

http://www.rutgers.edu/academics/financial-aid-tuition

http://www.rutgers.edu/info/information-job-

http://www.rutgers.edu/he

The Web-Interface (http://ec2-52-10-30-93.us-west-2.compute.amazonaws.com:8080/JaiMataDi16/)was developed using JSP. The usage of the Web-Interface is as follows -

1. The user enters the search term in the text box adjacent to the “Search” button as seen in the screenshot.

2. The search engine generates the output - the urls ranked using the algorithm explained later and displays them on the same page.

3. The flow of search will be explained later.

We share here the information about the Deployment Environment -

1. *Operating System* - Linux
2. *Location* - Cloud - Amazon AWS
3. *Java Environment* - Open JDK 6.0 (On the Server) - The application was developed using JDK 8 and binary was compiled to JDK 6
4. *Web Server* - *GlassFish 3.0 server* (during development GlassFish 4.1 and Netbeans IDE used)

4. Flow and Algorithm

We explain the flow and the algorithm together -

1. The user enters the search term - which can consist of one word or more than one word.

2. The textbox of the front-end accepts the search term and passes it via the JSP web application to the search engine

3. The search term is passed via the following java instruction in the environment-

   ```
   p = Runtime.getRuntime().exec("java -cp /home/ec2-user/project2/SearchEngine/bin
   searchEngineP.SearchEngine " + <search-term>);
   ```

   The above command calls the Environment (in this case the Linux operating system) and causes the operating system to execute the the SearchEngine.

4. The <search-term> contains the search-term entered by the user which it is passes as arguments to the SearchEngine

5. The main() method of the SearchEngine passes it to the externalSearch() method which is responsible for accepting input finding documents that contain the word or words of the search term using classes and various methods responsible for the inverted index, stemmed words and for ranking the output.

6. The algorithm inputs the words from the search-term and tries to find a minimumList. For search term with multiple words, minimumList contains those documents which contain *all the words in the search-term.* For single word search-term this is the list of all documents containing the single word being searched.

7. The algorithm handles the case where minimumList is empty - which means for multiple-word search-term that there is no single document in corpus which has all the words of the term.

8. It the collects a list of documents which has words (not necessarily all words in term) from the search-term. The ranking algorithm which is explained later will take care to try to rank such documents higher in the output which have most of the words of the search-term. If search-term has n words and all n do not occur in a document (which means that minimumList is empty), then the ranking algorithm will try to give higher preference to such documents which have n-1 words over documents with n-2 words from all the n words which constitute the search term.

9. *Intuition behind this - If it possible to find all the words which constitute a search term, in one document then rank such a document high in the output list , otherwise if it is not possible to find all words in one document , then the algorithm tries to rank those documents which have most words from the search-term higher in the output, over those documents which have fewer of the words of the search-term. For single word search-term the algorithm either*

finds all those documents in the corpus which contain that word. Thereafter the ranking is done in the following manner using the formula - .

Final Score of Document = 40% TFIDF + 60% DOCUMENT QUALITY (PRE-PROCESSED QUALITY SCORE)

Note - For multiple-words-search-term if there is no document in the corpus, which contains all the words in the search term, even though the algorithm tries to rank a document which has "most" words of the out of all the words (which constitute the multiple word search term), despite this fact it can not guarantee that a document with n-1 words (out of say n words of the multiple-word-search-term) would be ranked higher than a document with n-2 words. This is because 60% of the total score depends on the quality of the document, which is captured by the Document Quality score.

10. TF - IDF calculated using the Ranker class tfidfValue(String keyword, String fileName) method. The TF-IDF used follows the formula
    tfidf of a word ,w, is

    *tf(w) * idf(w)*

tf(w) - (number of times a word appears in a document) / (total number of words in document)
idf(w) - log(Number of documents/number of documents that contain word w)

11. Document Quality - Calculated using score(int docNumber) method of the Ranker class.

12. During Indexing we maintain two sets lowFrequencyWords and highFrequencyWords. They are stored in files for future usage.

13. lowFrequencyWords - These are words which occur just 1% in the entire corpus.
    highFrequencyWords - The remaining words which constitute the remaining 99% of the corpus.

14. After indexing is over, the scoring of all documents in the corpus is done as a part of pre-processing, and the scores are stored and are loaded to main memory in subsequent usage.

    Just like indexing is done once and stored so that during subsequent usage it can be loaded to main memory, these scores are also stored and later loaded into the main memory using a data structure - hash table which maintains the scores of all the documents of the corpus.

    The pre-processing of scores is done to save time and resources during subsequent usage. As we will explain below, the quality of document once fetched using crawler remains unchanged throughout its lifetime in the search engine, because it is the quality of the document as a whole and is completely independent of the search term.

The scoring is an expensive operation, since it is done once, when the application runs for the first time and is subsequently stored, the one-time cost is amortized over subsequent uses.

15. Intuition behind scoring (DQ) - words which occur less frequently in the corpus convey more semantic value than words which occur more frequently in the corpus.

16. A document with higher ratio of lowFrequencyWords / (total words in that document) conveys more semantic value in fewer words and therefore is considered as a higher quality document than a document which conveys same semantic value in more words i.e. which contains same number of lowFrequencyWords however, has higher total number of words Therefore if the *lowFrequency/Total word count* ratio is higher then the quality of the document is expected to be higher.

17. Thus both tfidf() and Document Quality is considered while scoring and determining final ranks of the URLs which would be displayed in the output to the user.

18. Then the documents are ranked from highest to lowest based on the score obtained from 40 % weight TFIDF + 60 % Document Quality

19. *The values 40% and 60% are obtained by empirical observation.*

20. Output is then displayed on the screen.

5. General Observations

1. When the user entered search term containing “,” the comma is converted to **%2C** in the browser and this degraded the performance and generated poor results.

2. Hence when the user enters comma - “,” or full stop “.” or hyphen “-“. These are removed from the search-term by the Web Interface before passing the arguments to the search engine.

3. *Since the previous versions the Indexing performance is better than earlier indexing, because the latter used to consume 10 minutes. After improvements, it used to take 90s and currently it takes just 10-15s to index all the parsed documents in the corpus.* ***The current indexing time is 40X to 60X times faster than its performance during first version.***

4. The Document Quality improved ranking over just using TFIDF alone to determine the final score of a document.

5. We believe that since Web-Interface calls the process - search engine through the environment this could be more amenable for cloud implementation of this project because in the cloud computing model, a organization is charged by cloud’s service provider, by the number of processors and memory used. As many SearchEngine processes will be launched as the number of users, which means the *usage of memory would grow linearly with number of users.*

6. The indices of the SearchEngine use HashMaps of Java and are efficiently algorithmically designed to be minimal in size (the size of index when on disk is just 1.1 MB for 687 web pages). This implies that for a search engine which has data of about 100,000 high quality web pages, the size of the index would be about 155 MB (assuming similar distribution of words as the given corpus of parsed documents.)

   Note - Currently even the commodity hardware computers have 8 GB RAM , therefore cloud computing is hoped to have better resources - on pay-as-you-use basis. Therefore this architecture should be scalable and would not be constrained by sockets.

7. It was found that nutch crawled several thousand links (see above for more details) but the output of nutch does not include parsed data for all the links crawled by it. When the dump.txt was visually inspected there were instances of pages that had certain fraction of the page’s data missing. This could be attributed to packet loss during transmission while fetching

8. The delay introduced in crawling is mainly due to the *robot.txt* files set by system administrators of the websites. This leads to slow crawling. It could be observed that during certain instances of crawling, the speed of fetching documents from a website could be as slow as *6 pages / minute*. This implies that, to crawl a web site with 100 web pages at the mentioned rate, it could take up to *17 minutes*. Considering there are thousands of websites, this is speed is slow.

9. *Technical Challenges* - We need to provide the path to the search engine. Initially, it was assumed that the search engine would be able to automatically find out its location in the deployment environment using getAbsolutePath() or getCanonicalPath() methods of File class of Java. However, this has not been successful in the deployment environment, though in development environment this worked as expected. *We think, that this should be due to the in-built security features of the Operating Systems offered by Amazon's cloud. This is because if a file (written in java or any other language) can automatically find its own location, then such an information puts it in a significant position to cause damage, if the file happens to be written for malicious purposes.* It is envisioned to overcome this challenge, by utilizing a separate configuration file for the search engine, which would be utilized to read the path to the search engine (as is the case with several commercial software).

Summary of Contributions

Phase 1 - Mrinal Monga - Understanding the nuances of nutch. Crawling and analyzing crawled data - which is used currently by search engine.

Riham S - Crawled and text processing

Phase 2 - Mrinal Monga - Designing and Developing the Search Engine and Indexer in Java.

Phase 3 - Mrinal Monga - Web-Interfacing - Designed and Developed the existing UI and inter facing used between UI and the Search Engine and Indexer in the Deployment Environment. Deployment of the Search Engine and Indexer and UI and interfacing in the Deployment Environment.

Riham S - Worked on a different UI.