Information Retrieval and Web Search Project

Project Theme: All US universities website crawler to retrieve nested links and body text.

Using Hadoop and Lucene

Team

Crawling Team	Specific Role
Prince Choudhary	Coding and determining structure, Debugging and solving exceptions, Monitoring and obtaining data.
Wijdan Alsurayya	Idea of using custom trust manager and implementation, Analysing data requirement, verification of data and deducing naming convention, File system structure design.

Indexing & SearchTeam	Specific Role
Varun Chawla	Lucene Index creation architecture design, Coding index creator, Debugging and file structure design.
Abhinav Gontu	Lucene Index code optimization, Quality Testing with large data, Requirement gathering and Analysis, System and Environment setup.

Wijdan Alsurayya

Abhinav Gontu

Varun Chawla

Prince Chaudhary

Overview of the crawling system

Theme: Crawl through the websites of all the universities in the USA and obtain data related to links and text.

Architecture:

Our crawler is built on java, we are using a few libraries for crawling and handling exceptions.

1. Libraries used: Jsoup, Junit4, java inbuilt libraries.
2. CrawlUniversities.java -> This has below two methods:

    1. main(): This is where our execution begins, it calls test() iteratively as per requirement.
    2. test(): It calls the crawl() method in class "EducationalDataCrawling" for crawling webpages.

3. EducationalDataCrawling.java -> This has two methods:

    1. prepareFilesandLinks(): Create link file, prepare queue for execution.
    2. crawl(): Crawl through webpages, handle exception, filter URLs, store data into files.

4. SslUtils.java -> It handles certificate issue when encountered.

Crawling Strategy:

1. Retrieve URLs of all the universities in the USA via from this page-http://doors.stanford.edu/~sr/universities.html ,this has links to 1000+ Universities.
2. We will filter and retrieve all the universities URLs and store them in Queue and process one by one.
3. We are only retrieving 'HTTP' pages so as to avoid retrieving any disallowed pages.
4. So as to justify our education theme, we perform breath first search and retrieve data for each university by limiting number of URLs visits to 50 (within this page or pages linked to it in chain), and we store them in files.
5. We have the freedom to change the limit of number of links we are crawling for each university. But we have chosen a number 50, as we want to have data which are more relevant. Increasing this limit could increase the possibility that data obtained is obscure.
6. We have used jsoup library to clean html pages so that we have have unnecessary data.
7. We are also checking if a website is already visited or not so that we don't retrieve same page multiple times or end up in loop, retrieving same set of pages again and again.

Instruction on how to deploy the crawler:

Step 1: Open file EducationalDataCrawling.java

Step 2: Edit pathname at all the 4 lines: Change it to the location at which you want to store the crawled data.

Step 3: Run main() at CrawlUniversities.java.

You are all set, crawling will start, you can track the progress on your console.

Obstacles and solutions:

1) While crawling we may encounter pages which are unreachable, server is not up or taking too much time to load. We have to have a strategy to handle this, as it is very time consuming and may also cause our program to stop.

Solution- We have used an innovative solution where we are running our crawler within a test block, and whenever we encounter a fault in our stack, our program would move on with the next iteration.

This is a very quick way to handle the situation, improving our performance significantly.

2)We may encounter "sslhandshakeexception" on some webpages as they mandate that we should communicate via information provided in certificate.

Solution: Created a custom trust manager to handle it.

Note:

- Make sure you have added JSoup and Junit4 libraries.
- We will modify it to make it more seamless in future.

Indexing Architecture:

Our Indexer is built on JAVA SDK, we are using Lucene Core and Query parser libraries in the dependencies like Analyzer, StandardAnalyzer, LongPoint, Document, StringField, TextField, IndexWriter, Directory.

Our IndexFile.java has 3 methods-

1. main(): This is where our execution begins, it calls indexDocs().
2. indexDocs(): It calls the indexDoc() method in class "IndexFile" for every document for indexing them.
3. indexDoc(): It indexes the document.

Indexing Strategy:

1. The indexer program will go to the path where the crawled documents are present which in this case is the *docsPath* present in the project folder.

2. If index folder is not present, then it will create the index file folder with the index path name present in the program.

3. We will be indexing only text data present in the HTTP pages via analyzer fields StringField, TextField, LongPoint .

Limitations:

The StandardAnalyzer does tokenizes URLs as tokens instead of URL

Fields used in Lucene index:-

StringField:-

There is no need to analyze the "pathField" field because we need to store the filename and search it after indexing. To reduce unnecessary matches of the filename we are using "StringField".

TextField:-

"contents" data of the html file is stored in "text" field. We are using "TextField" lucene field because we would need to analyze the content and then perform indexing so that when we search for a part of content, we still get significant results. It would be a waste of memory to store complete content after indexing therefore, we are using "InputStreamReader" which will take "InputStream" file pointer of the text contents of the html page. The content data is stored in the field temporarily only to index it after that the data is removed.

LongPoint:-

“Modified Time” data of the file is store in “lastModifiedTime” field using a “LongPoint” lucene field. It will store the latest modified timestamp of the crawled file.

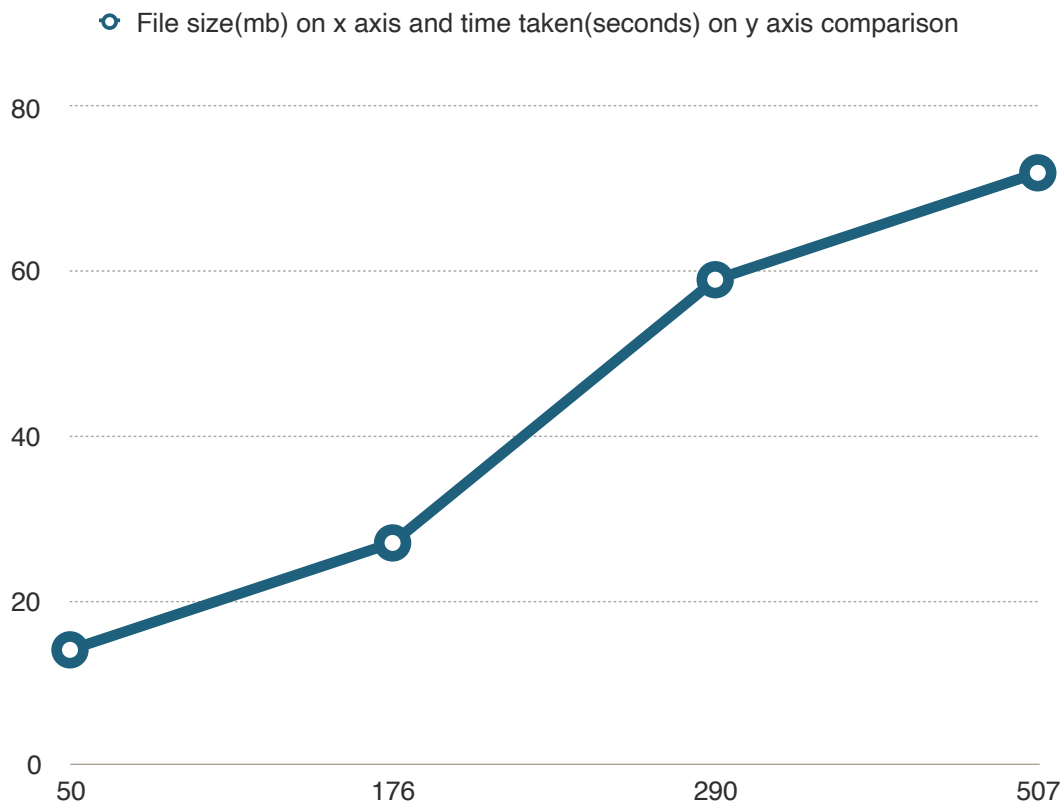
Analyzer choices:-

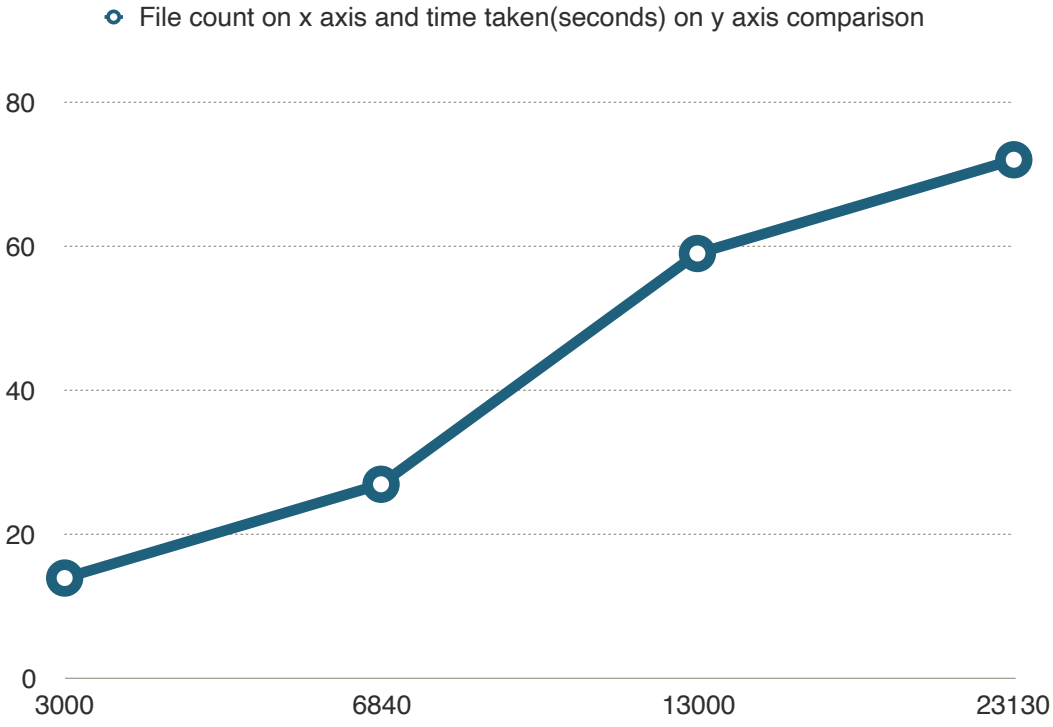
We are using a StandardAnalyzer because it will filter the stop words like "a", "the", "I", "be", "have", etc. . We can even add user defined stop words to the stop words set of StandardAnalyzer

For Crawling data and lucene indexing:

1) Go to the path where InformationRetrieval.jar is present
2) Make sure you have created two folders "docsPath" and "null", if not there already.
3) These folders will be used to store crawled data and lucene indexing files.
4) Run ./start.sh
5) You will be asked to enter the location where you want to store crawled data, enter- "docsPath"
6) You will be asked to enter the location where you want to store lucene index files, enter- "null"
7) You will be asked to enter the number of sublinks you wants to visit per universities, suppose enter- 20
8) You will be asked to enter the number of universities you want to visit, suppose enter - 10 (maximum 1015, even if you enter more it will crawl only 1015)
9) Once done the crawling will start automatically, and when crawling is done, indexer will automatically run.

Final Analysis of the Indexer:





Hadoop

Hadoop setup :-

1. Please need to change directory where hadoop is stored
$ cd /< directry where hadoop is stored >/hadoop-3.3.0/etc/hadoop

2. Open hadoop-env.sh file in order to set up the hadoop environment.
$ vi hadoop-env.sh
copy below line
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

3. Open hadoop-site.xml and add a property to it using below commands
$ vi hdfs-site.xml
add below line inside configuration tag

```
<property>
        <name>dfs.replication</name>
        <value>1</value>
</property>
```

4. Open core-site.xml and add a property to it using below commands
$vi core-site.xml
add below line inside configuration tag

```
<property>
      <name>fs.defaultFS</name>
        <value>hdfs://localhost:9000</value>
</property>
```

5. Open mapred-site.xml and add below properties to it using below commands

$vi mapred-site.xml

add below line inside configuration tag

```
<property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
</property>
<property>
    <name>mapreduce.application.classpath</name>
            <value>$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/*:
$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/lib/*</value>
</property>
```

6. Open yarn-site.xml and add below properties to it using below commands

$vi yarn-site.xml

add below line inside configuration tag

```
<property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
</property>
<property>
    <name>yarn.nodemanager.env-whitelist</name>

<value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_C
ONF_DIR,CLASSPATH_PREPEND_DISTCACHE,HADOOP_YARN_HOME,HADOOP_MA
PRED_HOME</value>
</property>
```

6. Hbase setup:-

Change to the conf directory in hbase in order to setup hbase

$ cd < directory where hbase is stored >/hbase-2.4.1/conf

$vi hbase-env.sh
Add the below line
export JAVA_HOME = /usr/lib/jvm/java-8-openjdk-amd64

$vi hbase-site.xml
add below line inside configuration tag

```
<property>
        <name>hbase.cluster.distributed</name>
        <value>false</value>
    </property>
    <property>
        <name>hbase.tmp.dir</name>
        <value>./tmp</value>
    </property>
    <property>
        <name>hbase.unsafe.stream.capability.enforce</name>
        <value>false</value>
    </property>
    <property>
        <name>hbase.zookeeper.property.clientPort</name>
        <value>2181</value>
    </property>
    <property>
        <name>hbase.rootdir</name>
        <value>file:///home/hadoopusr/HBASE/hbase</value>
    </property>
    <property>
        <name>hbase.zookeeper.property.dataDir</name>
```

```
        <value>/home/hadoopusr/HBASE/zookeeper</value>
    </property>
```

After this we need to change bashrc file

$ vi ~/.bashrc

copy below lines

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_HOME=/usr/local/bigdata/hadoop-3.3.0
export PATH=$PATH:$HADOOP_HOME/bin
export HBASE_HOME=/usr/local/bigdata/hbase-2.4.1
export PATH=$PATH:$HBASE_HOME/bin
```

$ source .bashrc

SSH:-

$ssh localhost

If can't connect to ssh then execute below commands

```
$ ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
$ chmod 0600 ~/.ssh/authorized_keys
```

Preprocessing steps:-

The below steps needs to be performed before running any map reduce jobs
We can start Hadoop by below command
$ ./< directory where hadoop is stored >/hadoop-3.3.0/sbin/start-all.sh

We can start hbase by below command
$ ./< directory where hadoop is stored >/hbase-2.4.0/bin/start-hbase.sh

change to the directory where the pom.xml file of indexer part of the project is present.

$ cd /< directory where the project is stored >/project/HbaseIndex/searchmr

The below command is used to build the mapreduce jar file which will be used to perform indexing.

$ mvn clean compile package assembly:single

The below command is used to index files and we will run it only once.

$ hadoop jar /< directory where the project is stored >/project/HbaseIndex/searchmr/target/HbaseIndex-1.0.0-jar-with-dependencies.jar edu.ucr.abhi.index.Indexer /<HDFS directory where crawled data is stored>/*

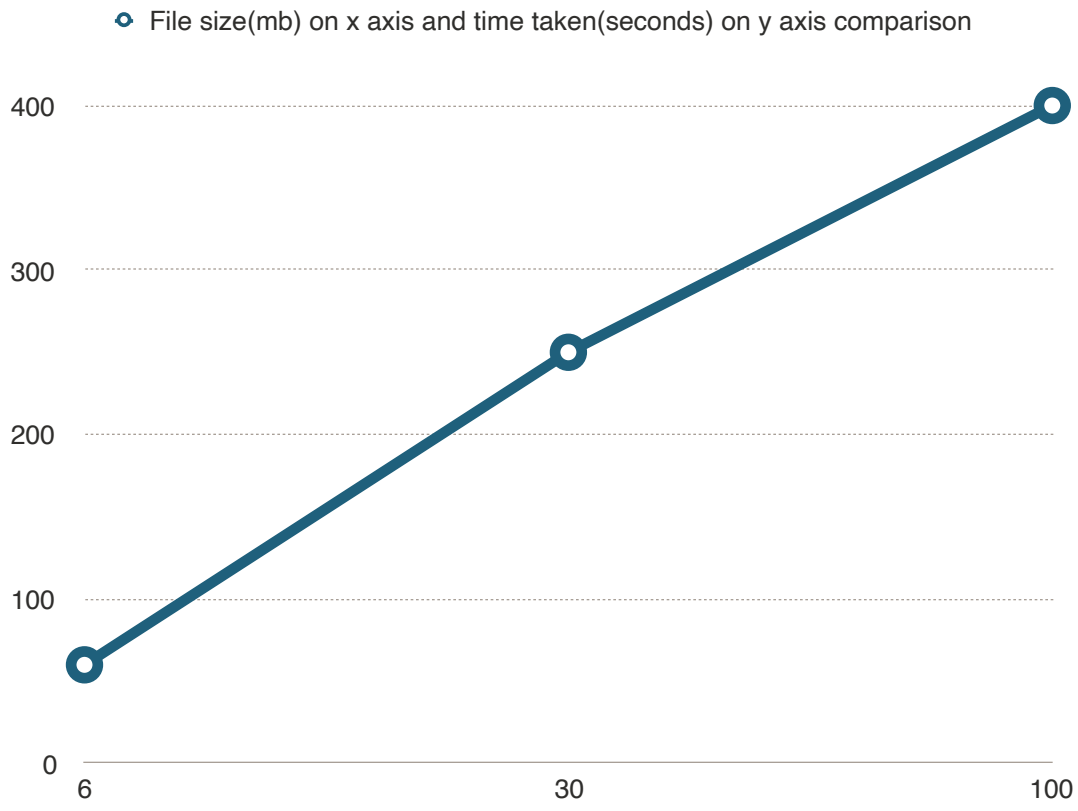
PLEASE NOTE: IN ORDER TO RUN SEARCHING THROUGH HADOOP YOU WILL HAVE TO DO RUN HADOOP INDEXING BY FOLLOWING THE ABOVE PROCEDURE TO GET RESULTS IN THE HADOOP QUERY.

How Hadoop was used key, value definitions and data flow:

Basically we will be using postings data structure in which the key will be tokens and there will be multiple values like href, title, count etc. In the indexing we have two modes of operations with and without href. So, all the postings are aggregated in json format in json messages, so once the whole indexing part is done we have ranking where we pass query and we get sorted urls based on their counts. Hadoop has inherent proper in the reducer phases map reducer job output is sorted by keys, so we set inverse count as key.

Whenever we get a word, it is associated with a posting, this is in the map phase, in the reducer phase, the same token associated with multiple postings, we combine those postings into kind of map like data structure and then we searilze these to the JSON and then put them in hbase. The entity for which the count is maximum, we are displaying it on the top and giving it the highest score which is the basis of our score in indexing.

Hadoop Graph:-



Web Interface

The Web Interface was made in Bootstrap, HTML, CSS and Javascript and was integrated into SprinBoot with Thymeleaf in Java. The Web Interface was enabled with with Speech Recognition and the screenshots are as below-

Lucene Results:-

Number	Title	URL	Score
1	Alabama A&M University - Alabama A&M University	http://www.aamu.edu	0.7444998
2	Archives and Special Collections	http://www.lib.alasu.edu/archives	0.70666844
3	WJAB	Your Smooth Jazz and Cool Vocals station. Powered by Alabama A&M University	http://www.wjab.org	0.7046939
4	Levi Watkins Learning Center	http://www.lib.alasu.edu	0.6815914
5	Home	Alabama State University	http://www.alasu.edu	0.64897895
6	Levi Watkins Learning Center	http://www.lib.alasu.edu/library/departments/media	0.52083004
7	J. F. Drake Memorial LRC	http://alabamam.sdp.sirsi.net	0.4438092

Want to search more? Go Back.....

localhost:8080/results#

Hadoop Results:-

Number	Title	URL	Score
1	Home | Alabama State Universitya A&M Universityn Long Island, NYnta A&M University	www.alasu.edugift.orgtalssolutions.com	25.0
2	Levi Watkins Learning Centervices - LibGuides at Alabama A & M Universitysityrsity	www.lib.alasu.eduumnetalssolutions.com	10.0
3	Alabama A&M University - Alabama A&M Universityn Long Island, NYnta A&M University	www.aamu.eduugift.orgtalssolutions.com	9.0
4	WJAB | Your Smooth Jazz and Cool Vocals station. Powered by Alabama A&M University	www.wjab.orgmu.eduorgtalssolutions.com	9.0
5	Alabama A&M University - Alabama A&M University. Powered by Alabama A&M University	www.aamu.eduu.eduuorgtalssolutions.com	9.0
6	Levi Watkins Learning CenterCool Vocals station. Powered by Alabama A&M University	www.lib.alasu.eduuorgtalssolutions.com	5.0
7	Archives and Special Collections A&M University. Powered by Alabama A&M University	www.lib.alasu.eduuorgtalssolutions.com	4.0
8	The National Center for the Study of Civil Rights and African American Culturesity	www.lib.alasu.edu.orgtalssolutions.com	4.0
9	Circulation - LRC Access Services - LibGuides at Alabama A & M Universitysityrsity	libguides.aamu.edumnetalssolutions.com	3.0
10	Home - LibGuides at Alabama State Universityuides at Alabama A & M Universityrsity	alasu.libguides.comnetalssolutions.com	3.0
11	Course Reserves - LRC Access Services - LibGuides at Alabama A & M Universityrsity	libguides.aamu.edu.netalssolutions.com	3.0
12	A-Z Databasesmooth Jazz and Cool Vocals station. Powered by Alabama A&M University	libguides.aamu.eduorgtalssolutions.com	2.0
13	Home - LibGuides at Alabama A & M Universityights and African American Culturesity	libguides.aamu.eduorgtalssolutions.com	1.0
14	J. F. Drake Memorial LRClections A&M University. Powered by Alabama A&M University	alabamam.sdp.sirsi.netalssolutions.com	1.0
15	Home - Interlibrary Loan - LibGuides at Alabama A & M Universityniversitysityrsity	libguides.aamu.edumnetalssolutions.com	1.0

Want to search more? Go Back.....

Notes:-

1. As perviously suggested by Luxun, we have indexed hyperlinks and implemented bottleneck to not lose put pressure on memory.
2. Make sure you have have added Lucene libraries from https://lucene.apache.org/
3. We will modify it to make it more seamless in future.

References:-

1) https://jsoup.org/
2) https://www.programmersought.com/article/23935356341/
3) https://lucene.apache.org/
4) Class lectures, slides and notes

Link to 1.2 gb file crawled:-

https://drive.google.com/file/d/1wN6MLIdJ1n3gL9vK-s42rdC1JZ3Bu1xy/view?usp=sharing

You can use the below files which are converted into single line to run Hadoop Indexing

https://drive.google.com/file/d/1E97UBH7GcU14WQ4yv688n1fTFSzRHyby/view?usp=sharing

https://drive.google.com/file/d/1TKaG_XBtRrO2OffEWqJMlmrckuOUW_qw/view?usp=sharing

https://drive.google.com/file/d/1AwqzOLuP8F893EPuoc6mxSz3uQfDWeBF/view?usp=sharing

We would like to thank Professor Vagelis Hristidis for giving us this exciting and fruitful opportunity to work on this integrated project so that four people from different cultures, knowledge, backgrounds and skillsets could come together and perform this part of the the project. We would also like to thank Luxun Xu for giving us the demo tutorials on how to execute this part of the project. Thank you!