CIS555 Project README

1) Full names of project members: Joop Hong (joophong), Kevin Lee (kevlee), Rohit Gupta (guptaro), Yunchen Wei (yunchenw)

2) Features implemented:
   a) **Multi-node, distributed crawler**: Crawls html, xml, txt and pdf files. Also retrieves Alexa site rankings for each site crawled (separate component). Filters out the majority of non-English sites (as well as sites with little English content).
   b) **Indexer**: Different indexers for html, xml, txt and pdf files. Created inverted indexes for documents, including hit position, fancy hit information (e.g. is the word in title / anchor text / is capitalied / in img alt text?). Only indexing words with ascii characters.
   c) **PageRank calculator**: Takes files that have link structure, and computes PageRank through multiple iterations of MapReduce. Uses decay factor of 0.85 and removes sinks as per original Google PageRank paper.
   d) **Search Engine**: Uses weighted sum of the following factors to rank documents (URLs) that contain at least one of the words in the query: i) Okapi BM25 value that is computed based on tf and idf values, ii) Pagerank value, iii) The degree of relevance based on word positional information, iv) Alexa site rankings and v) Whether or not there is a fancy hit

3) Extra credit claimed:
   a) Support crawling additional content types: pdf, txt, xml
   b) Show preview (small excerpt) of hitting pages along with search results
   c) AJAX support for auto-completion
   d) Integrate search results with Alexa ranking for better ranking
   e) Integrate search results from Wikipedia, e.g. show intro part of wikipedia search
   f) Items not crawled but linked by crawled files (eg .jpg files) are also included as search results

4) Source files

```
├── src
│   ├── cis555
│   │   ├── PageRank
│   │   │   ├── PageRankCleanMapper.java
│   │   │   ├── PageRankCleanReducer.java
│   │   │   ├── PageRankDriver.java
│   │   │   ├── PageRankInitMapper.java
│   │   │   ├── PageRankInitReducer.java
│   │   │   ├── PageRankMapper.java
│   │   │   ├── PageRankReducer.java
│   │   │   └── utils
│   │   │       ├── CountDriver.java
│   │   │       ├── CountMapper.java
│   │   │       ├── CountReducer.java
│   │   │       ├── SortCombineMapper.java
│   │   │       ├── SortCombineReducer.java
│   │   │       ├── SortDriver.java
│   │   │       ├── SortMapper.java
│   │   │       └── SortReducer.java
│   │   ├── aws
│   │   │   └── utils
│   │   │       ├── AWSClientAdapters.java
│   │   │       ├── AWSConstants.java
│   │   │       └── S3Adapter.java
│   │   ├── contentTransferer
│   │   │   ├── ContentTransferer.java
│   │   │   └── PopulateS3.java
│   │   ├── crawler
│   │   │   ├── Crawler.java
│   │   │   ├── CrawlerException.java
│   │   │   ├── CrawlerUtils.java
│   │   │   ├── GETWorker.java
│   │   │   ├── LinkExtractorWorker.java
│   │   │   ├── LinkQueuer.java
│   │   │   ├── RawCrawledItem.java
│   │   │   ├── Response.java
│   │   │   ├── ResponseException.java
│   │   │   ├── RobotsException.java
│   │   │   ├── RobotsMatcher.java
│   │   │   ├── SiteInfo.java
│   │   │   └── ValidationException.java
│   │   ├── database
│   │   │   ├── CrawlerDao.java
│   │   │   ├── EntryDoesNotExistException.java
│   │   │   └── UnauthorisedException.java
│   │   ├── indexer
│   │   │   ├── DocHit.java
│   │   │   ├── DocHitObject.java
│   │   │   ├── HTMLIndexer.java
│   │   │   ├── Indexer.java
│   │   │   ├── PDFIndexer.java
│   │   │   ├── TXTIndexer.java
│   │   │   ├── XMLIndexer.java
│   │   │   └── mapreduce
│   │   │       ├── CombineFileInputFormatImpl.java
│   │   │       ├── CombineFileRecordReaderImpl.java
│   │   │       ├── IndexerDriver.java
│   │   │       ├── IndexerMapper.java
│   │   │       ├── IndexerReducer.java
│   │   │       ├── URLTypeWritable.java
│   │   │       ├── WholeFileInputFormat.java
│   │   │       └── WholeFileRecordReader.java
│   │   ├── pingAlexa
│   │   │   └── PingAlexa.java
│   │   ├── searchengine
│   │   │   ├── AjaxRequestServlet.java
│   │   │   ├── ContentDAO.java
│   │   │   ├── FetchAndPopulateScript.java
│   │   │   ├── IndexTermDAO.java
│   │   │   ├── MainServlet.java
│   │   │   ├── MyServletContextListener.java
│   │   │   ├── PagerankDAO.java
│   │   │   ├── QueryProcessor.java
│   │   │   ├── RankServlet.java
│   │   │   ├── ServletHelper.java
│   │   │   ├── ShutdownHook.java
│   │   │   ├── ShutdownServlet.java
│   │   │   ├── Test.java
│   │   │   ├── UrlIndexDAO.java
│   │   │   └── utils
│   │   │       ├── DocHitEntity.java
│   │   │       ├── DocIDContentInfo.java
│   │   │       ├── DocIdPagerankInfo.java
│   │   │       ├── IndexTerm.java
│   │   │       ├── QueryTerm.java
│   │   │       ├── SEHelper.java
│   │   │       ├── WeightedDocID.java
│   │   │       ├── WordWithPosition.java
│   │   │       └── WordWrapper.java
│   │   ├── urlDispatcher
│   │   │   ├── master
│   │   │   │   ├── MasterServlet.java
│   │   │   │   ├── StatusPageContentGenerator.java
│   │   │   │   └── WorkerDetails.java
│   │   │   ├── utils
│   │   │   │   ├── DispatcherConstants.java
│   │   │   │   ├── DispatcherException.java
│   │   │   │   └── DispatcherUtils.java
│   │   │   └── worker
│   │   │       └── WorkerServlet.java
│   │   └── utils
│   │       ├── CrawledDocument.java
│   │       ├── CrawlerConstants.java
│   │       ├── DBException.java
│   │       ├── DBWrapper.java
│   │       ├── DocIdUrlInfo.java
│   │       ├── DocumentMeta.java
│   │       ├── FastTokenizer.java
│   │       ├── FromToUrls.java
│   │       ├── Hit.java
│   │       ├── Stemmer.java
│   │       ├── UrlDocIDMapper.java
│   │       └── Utils.java
```

(Please note that we have not included any Hadoop-related libraries in the project lib folder because they are extremely big and numerous).

5) How to install and run the project

Crawler

1. Update the master's IP address in conf/workerWeb.xml (under param name 'master'), then run ant-build.
2. For each crawler, start a new t2.medium instance, and copy for_ec2.zip to an EC2 instance (use at least a t2.medium instance).
3. Unzip for_ec2.zip, and chmod the ec2_setup_scripts directory
4. Run enviornment_setup_on_ec2.sh to install the relevant libraries, and then run setup_on_ec2.sh to create the necessary directories
5. For the master node, run restart_on_ec2.sh. For the other crawler nodes, run worker_only_on_ec2.sh
6. Go to <master's ip address>:8080/master/status, and once all the other workers appear on the site, click the Start button to start the crawl.
7. The crawled files are stored in crawler_db/crawled_files
8. The meta information for each viewed URL and links are stored in the database. To extract them to a file, from your host computer copy transferer.jar to each EC2 instance, and run transferer.jar (java -jar transferer.jar). This will extract all the contents to crawler_db/document_meta and crawler_db/crawled_urls, which can be subsequently sent to s3.

9. In order to get Alexa Rankings for each site crawled, upload alexa.jar to each EC2 instance, and run it. The resulting Alexa Rankings will be stored in crawler_db/alexa

Indexer:

To save space, as well as reduce the transmitting time for documents, input file for indexer is in the compressed version, with the file name as [DOC_ID].[DOCTYPE].gzip (the gzip file starts with 400 byte containing the URL for given docID.)

1. Compile packages cis555.indexer, cis555.indexer.mapreduce, cis555.utils, as well as required libraries in lib/* into a jar file, with cis555.indexer.mapreduce.IndexerDriver as the main application entry point
2. Set up the input directory and output directory as pass-in arguments, in the form of [input_dir] [output_dir]
3. To run on EMR, upload the jar file into S3 bucket, add the jar file as a step, set input directory and output directory as arguments.

Page Rank Calculator

Note your input files must have the following format. Each docID has the format of a 32-character ID (only hex characters allowed). Each line of the input files must be docID of outgoing URL (call this docID_out), then a tab, then docIDs of the URLs that docID_out links to

1. Compile src/cis555/PageRank/* with JARs in your class path: hadoop-common-2.6.0.jar and hadoop-mapreduce-client-core-2.6.0.jar
2. Use these class files to make a JAR
3. Set up your input and intermediate directory. Say your input files live in INPUTDIR. Then create a directory called INTERMEDIATEDIR which is empty, which is where all the iterations will be stored. Choose your OUTPUTDIR but don't create it (Hadoop does not like output directory to be created). Note you should do all this regardless of whether you want to run locally or on EMR
4. To run locally
    a. Run on command line: export JAVA_HOME=your_Java_runtime_directory, where your_Java_runtime has the bin directory in which your Java runtime lives
    b. Run this on the command line: hadoop-2.6.0/bin/hadoop jar $1 $2 $3 $4 $5
    c. $1 = name of JAR file
    d. $2 = name of main class, in this case should be cis555.PageRank.PageRankDriver
    e. $3 = INPUTDIR
    f. $4 = INTERMEDIATEDIR
    g. $5 = OUTPUTDIR

5. To run on AWS
    a. Start an EMR cluster. The default settings are fine.
    b. Upload the JAR to S3.
    c. Run a step on your cluster (custom JAR).
        i. Main class: cis555.PageRank.PageRankDriver
        ii. Arguments: INPUTDIR INTERMEDIATEDIR OUTPUTDIR

Search Engine:

1. All relevant data for processing the user query are stored locally in the EC2 instance that hosts the web application.
2. To fetch raw data from S3 to EC2 and populate BerkeleyDB based on the data, export the entire project as a Runnable JAR with cis555.searchengine.FetchAndPopulate set as the entry point.
3. Run the JAR file in /usr/inputdata of EC2. This will copy the data from S3 to /usr/inputdata/S3DATA and build 4 databases--ContentDB, IndexTermDB, PagerankDB, and UrlIndexDB-- in /usr/inputdata/database.
4. To build the web application into a WAR file, run the following ant build command:
    ```
    [ant -buildfile searchengine_build.xml all]
    ```
    For a successful build, web.xml must be in /target/WEB-INF.
5. To deploy the WAR file, one can either scp the file to to /usr/share/tomcat7/webapps directory in the EC2 instance or use the tomcat web manager UI to upload the file. Restart Tomcat by the following command:
    ```
    [sudo service tomcat7 restart]
    ```