CS47300 Project-1 (updated 09/14/17)

1. Download and setup Lemur

Follow instruction to install the Lemur system (Instructions are given for Linux. If you don't have the system available, you can use the CS department's mc server.):

- Install maven (if you already have, you can skip this)

    1. Download maven
        a. wget http://mirror.olnevhost.net/pub/apache/maven/binaries/apache-maven-3.2.1-bin.tar.gz
    2. Run the following to extract the tar:
        a. tar xvf apache-maven-3.2.1-bin.tar.gz
    3. Next add the env variables such as
        a. export M2_HOME= <your_maven_path>/apache-maven-3.2.1
        b. export M2=$M2_HOME/bin
        c. export PATH=$M2:$PATH

- Install galago

    1. Download Lemur from: https://www.lemurproject.org/
    2. Extract galago-3.12.tar.gz file to a separate folder
    3. Make sure that, Java JDK 1.8.0 or higher and Maven 3.0 or higher is installed on your Linux system. If not, install them before proceeding further.
    4. Change directory to galago-3.12 folder:
       E.g. cd galago-3.12/
    5. ./scripts/installlib.sh
    6. mvn package
    7. mvn install
    8. chmod +x ./core/target/appassembler/bin/galago

Instructions adapted from:
https://sourceforge.net/p/lemur/wiki/Galago:%20Quick%20Guide%20for%20New%20Users/

2. Building Indices (6 pts)

Download the **Wiki Small** (6043 documents) [tar.gz, 26 MB] from the book's webpage:
http://www.search-engines-book.com/collections/

a) Build an index locally with default parameters (this uses a krovetz stemmer).

E.g. ./core/target/appassembler/bin/galago build --indexPath=wiki-small-index --inputPath=wiki-small

b) Start an interactive web server.
E.g. ./core/target/appassembler/bin/galago search --index=wiki-small-index --corpus=wiki-small

c) Report the top three documents returned for the queries “probabilistic analysis” and “decision model". **(3 pts)**

d) Build another index without stemming and query without stop word removal (by changing the configuration files and parameters). This index will be used for further exercises in this project. Include your command / configuration in your report. **(3 pts)**

3. Accessing the Index (24 pts)

Write a program to compute the following statistics based on the index you built in 2(d)

a) The total number of documents in the corpus and the average length of the documents. **(4 pts)**
b) The number of unique words appeared in the whole corpus. **(4 pts)**
c) Find the longest document(s). Report the document number(s) and their length(s). **(2 pts)**
d) Report the number of documents containing the words “probabilistic", “analysis ", “decision", and “model". **(4 pts)**
e) Compute and report the IDF of the four words: “probabilistic", “analysis ", “decision", and “model” **(2 pts)**
f) Determine documents with the maximum number of occurrences of “probabilistic" and “model". Compute the TF-IDF scores for these documents. **(4 pts)**
g) Determine the documents with maximum occurrences of the words “entropy" and “science". Compute the Okapi scores (https://en.wikipedia.org/wiki/Okapi_BM25) for these documents. (Use $k_1 = 1.2$, $b = 0.75$)

$$\text{score}(D, Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{\text{avgdl}})}$$

Where $f(q_i, D)$ is q_i’s term frequency in document D, $|D|$ is length of document D in words, and avgdl is the average document length in the text collection from which documents are drawn.

Calculate IDF(q_i) using following equation:

$$IDF(q_i) = \log \frac{N - n(qi) + 0.5}{n(qi) + 0.5}$$

Where N is the total number of documents in the collection, and $n(q_i)$ is the number of document containing word q_i.

Include these statistics in your report. Submit your java source code along with your report. **(4 pts)**

Suggestion

- If you haven't figured out how to import the compiled classes to your code, you can download the following jar file and add it your classpath.
  - http://www.cs.purdue.edu/~clifton/cs47300/Project1Jarfiles.zip
  - Unzip the file and use it when you compile and run your code
    E.g. javac -cp "galago_cs473.jar" testcode.java
    java -cp "galago_cs473.jar:.:lib/*" testcode
    If you are running on CS department machines, you can just 'ln –s /homes/clifton/cs47300/jars/* .' to avoid making copies
    (or change the classpath to include the appropriate files/directories from the above.)
- Otherwise, you can also create an Eclipse project using maven using the command below.
  - mvn -DdownloadSources=true eclipse:eclipse
  - Then, add your code to the created project

4. Evaluation (10 pts)

Download the **CACM** (3024 documents) [tar.gz, 1MB] [relevance judgments] [processed queries] from the book's webpage:

http://www.search-engines-book.com/collections/

a) Build an index for the **CACM** corpus locally with default parameters.
b) Construct a json-query consisting of queries 4, 7, 16, and 18 from the downloaded "processed queries". Use the galago batch-search command with the json-query to output the **Top-100 documents** to use below. Report the **Top-2 documents** for each query. **(5 pts)**
c) Evaluate both the IR models from above (TF-IDF / Okapi) using Mean Average Precision (MAP), Normalized Discounted Cumulative Gain (NDCG), and Precision@20 for each query using the **Top-100 documents** you extracted above. **(5 pts)**

5. How to Turn in

- Compress your report (pdf) and code (just the source, not the entire directory/compiled files/etc.) into a file
- Submit the file to blackboard