Lab 9 – Ranked Search over Wikipedia

CC5212-1

May 27, 2015

Today we combine the last two labs: we use the PageRank scores computed in the last lab to improve the results of the search engine we built two labs ago. Let's see if the results improve with PageRank, eh?

- The first step is to get the PageRank scores of all Wikipedia articles and decode them. (If you already have the decoded ranks for all of Wikipedia, then you can skip ahead.) For this, you will need working code for Lab 8 (the completed `mdp-lab8` package; if you do not have this, please talk to me).
  - If you do not have the raw data, you can download it from `http://aidanhogan.com/teaching/cc5212-1/data/lab8/es-wiki-links.gz`.
  - We first need to compress the data, converting URLs to integer IDs. Call `OIDCompress` with:[1]

    ```
    -i [dir]/es-wiki-links.txt.gz -igz -o [dir]/es-wiki-links.oid.txt.gz -ogz
     -d [dir]/es-wiki-links.dict.txt.gz -dgz
    ```

  - Run your working copy of `PageRankGraph.java` over the encoded file:

    ```
    -i [dir]/es-wiki-links.oid.txt.gz -igz -o [dir]/es-wiki-ranks.oid.txt.gz -ogz
    ```

  - Next run `SortByRank` for the output of the big file. This will order the documents by rank.[2]

    ```
    -i [dir]/es-wiki-ranks.oid.txt.gz -igz -o [dir]/es-wiki-ranks.s.oid.txt.gz
     -ogz
    ```

  - Time to convert the integer IDs for articles back to URLs. Call `OIDDecompress`:

    ```
    -i [dir]/es-wiki-ranks.s.oid.txt.gz -igz -o [dir]/es-wiki-ranks.s.txt.gz -ogz
     -n 0 -d [dir]/es-wiki-links.dict.txt.gz -dgz
    ```

  - Decompress and open up the output file if curious which articles are highest ranked. ☺
- The next step is to open up the code for Lab 7 (package `mdp-lab7`; if you do not have the code, you can grab it from `http://aidanhogan.com/teaching/cc5212-1/code/mdp-lab7-sol.zip`). We want to build the inverted index (if you already have an inverted index built on your computer, you can skip this step).
  - Grab the data from `http://aidanhogan.com/teaching/cc5212-1/data/lab7/es-wiki-abstracts.tsv.gz` if you don't have it already. This contains the URL, title and abstract of all Wikipedia articles.
  - Run the class `IndexTitleAndAbstract` with

    ```
    -i [dir]/es-wiki-abstracts.tsv.gz -igz -o [output-dir]
    ```

  - Let's collect some search results before adding the PageRank scores to see if the results improve:
    - Run the class `SearchIndex` with the argument `-i [dir]`.
    - Run some searches and copy the results into a text file and save them for comparison later: search for "`obama`", "`boston`" and "`neruda`" and three other searches of your choice.[3]

[1] If you have trouble with memory, add, e.g., `-Xmx1500M` to the VM arguments to increase the heap.
[2] Actually this part is not really necessary ... but is interesting out of curiosity to see the top-ranked articles.
[3] You may encounter some problems for searches involving accents. I haven't managed to figure this out yet. :(

.

- Last but not least, we want to use the PageRanks to increase the score of more important articles in Wikipedia (based on their link structure).
  - First make a copy of the inverted index directory. We will include the ranks in this copy.
  - Import `http://aidanhogan.com/teaching/cc5212-1/code/mdp-lab9.zip` into Eclipse.
  - You need to code the `boostRanks(.,.)` method. Look at the `IndexTitleAndAbstract` class and the `SearchIndex` classes for examples (lab 7 code).
    - First open a `Directory` over `indexDir` and then create a `StandardAnalyzer` with version `LUCENE_48` (see `IndexTitleAndAbstract`).
    - Open an `IndexWriterConfig` as before, but this time open it in append mode, not create mode (since we just want to modify entries, not create them) (see `IndexTitleAndAbstract`).
    - Create an `IndexWriter` (see `IndexTitleAndAbstract`).
    - Next you want to open an `IndexReader` over `indexdir` (see `SearchIndex`).
    - Now you need to iterate over all of the documents in the index and add the boost based on the rank.
      - The number of documents in the index you can find by calling `.maxDoc()` on the `IndexReader` object you created.
      - You can fetch a specific document (for $0 \geq$ `i` $>$ `maxDoc()`) by calling `.document(i)` on the reader object.
      - Call `.getField(FieldNames.URL.name()).stringValue();` on the document object to get its URL.
      - Get the rank of the article based on its URL from the map. Make sure to check that the rank is not null. If it is, set the rank to 0.
      - We do not want to use the raw ranks directly as a boost score since they are very small and we do not want boosts of 0. Instead pass the rank through `getBoost` (see bottom of `BoostRanks` class) to get the boost score (a float). (Note: the method is quite arbitrary.)
      - Let's index the rank of the document. To do this, create a new `DoubleField` – with name `Field.RANK.name()`, raw rank score and `Store` set to yes – and add it to the document.
      - Let's boost the score of the title field using the rank value (where `doc` is the document):

```
IndexableField title = doc.getField(FieldNames.TITLE.name());
((Field)title).setBoost(boost);
```

      - Finally, let's update the document in the index:

```
Term urlT = new Term(FieldNames.URL.name(),url);
writer.updateDocument(urlT,doc);
```

      - Don't forget to print a message for every `TICKS`$^{\text{th}}$ document processed.
    - Don't forget to close the writer at the end of the method.
  - Time to run BoostRanks over the index:

```
-i [dir]/es-wiki-ranks.s.txt.gz -igz -o [index-dir-copy]
```

- The last item: pass the new search index to `SearchIndex` and run the same six searches again. Are the results better? (Optional: modify `SearchIndex` to print the rank when available.)
- **IMPORTANT:** Submit `PageRankGraph` to **lab 8** on U-Cursos (only if you completed it yourself). Submit `BoostRanks` to **lab 9**.