Project Part 2

Due November 21, 2017, 11:59PM on Blackboard

The goal of this part is to read a small collection of movie reviews, and to store on disk an index and a document table.

The indexing program will read the files stored in reviews*.rar* on Blackboard and create an index.

The index will include a dictionary of all the indexed terms, and the postings lists for each. Each record of the dictionary will contain, the term, the number of documents containing the term (df), and the pointer to the posting file for the term.

Each record in a posting list will include a docID. Use your code of part1 to extract index terms.

In addition to the index you will also generate a document table. This table will be used by the retrieval program to display the retrieval results. The document table will include:

1. File name
2. Title
3. Reviewer
4. Snippet
5. Rate (P for positive and N for negative, NA for no answer)

There are many lists of positive and negative terms on the Internet. A possible one can be found in <http://www.wjh.harvard.edu/%7Einquirer/spreadsheet_guide.htm>

Creating a static snippet

1. Use the text in the capsule review if the document contains one as the snippet. If it is short add text from the beginning of the review for a total of 50 words.
2. Otherwise, use the first 50 words of the review, or add more heuristics to deal with the problem.

Assigning Rates

1. Search for the text fragment “-4 to +4 scale” and look for a number preceding the fragment. If the number assigned by the reviewer before this fragment of text is >=0 store P for the review rate, otherwise store N.
2. Otherwise, if the review contains “CAPSULE REVIEW”, or “capsule” count positive words such as best, exciting, outstanding in the capsule and subtract the count of negative words such as dull, boring, disappointing, failure etc. Store P if the result is >=0, otherwise N.
3. If the text contains “rate” followed by the name of the movie, and the following text before the end of the sentence contains positive terms store P else if it contains negative terms store N.
4. Otherwise the problem becomes harder. Store NA or add heuristics to get a better answer.

The input parameter to the program is:

A directory path name

The output files will be:

1. Dictionary.csv
2. Postings.csv
3. DocsTable.txt

Each field in the csv files will be separated by a comma.

EXAMPLE

There are 2 documents 1 and 2.

Doc1 title = Titanic

Doc 2 title = Casablanca

Doc 1 body = Titanic movie. Titanic disaster. Titanic boat.

Doc 2 body = German occupation movie.

The doc table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Doc number | Title | Reviewer | Snippet | Rate |
| 1 | Titanic | x |  | P |
| 2 | Casablanca | y |  |  |

The dictionary The posting file

|  |  |  |
| --- | --- | --- |
| Term | df | offset |
| boat | 1 | 0 |
| casablanca | 1 | 1 |
| disaster | 1 | 2 |
| german | 1 | 3 |
| movie | 2 | 4 |
| occupation | 1 | 6 |
| titanic | 1 | 7 |

|  |  |
| --- | --- |
| DocID | tf |
| 1 | 1 |
| 2 | 1 |
| 1 | 1 |
| 2 | 1 |
| 1 | 1 |
| 2 | 1 |
| 2 | 1 |
| 1 | 4 |

**Submission directions**

As in part 1.

**Plagiarism Policy**

All your code will be checked for similarity to other submissions using Moss. Programmers have an uncanny knack of reproducing the same code that they have seen before. So you are advised not to look at each other's code.