INFO 4300: HW I

Jason Marcell

September 19th, 2010

1 Introduction

The object of this assignment was to write a program called invert that takes raw text, creates an inverted file system, and provides a simple user interface for testing. A corpus of 30 documents from Reuters and a stop list were provided.

2 Implementation

2.1 Data-Structures

Every word from every document from the corpus is placed into a hashtable where the key is the word and the value is yet another hash table. The key of the second hash table is the document name and the value of the second hash table is a dynamic array of integers representing the locations in the document where instances of the word were found.

```
1  ["aaron", {"./test/file25.txt"=>[626]}]
2  ["abandoning", {"./test/file22.txt"=>[2594]}]
3  ["ability",
4   {"./test/file27.txt"=>[3901],
5    "./test/file23.txt"=>[2363],
6    "./test/file13.txt"=>[1896]}]
7  ["able",
8   {"./test/file27.txt"=>[2496],
9    "./test/file17.txt"=>[2177],
10   "./test/file28.txt"=>[864, 3367],
11   "./test/file11.txt"=>[1728, 2984],
12   "./test/file15.txt"=>[3238],
13   "./test/file24.txt"=>[195]}]
14 ["aboard",
15  {"./test/file09.txt"=>[1952, 2144],
16   "./test/file10.txt"=>[517, 1759],
17   "./test/file03.txt"=>[538, 1570]}]
```

Listing 1: A small sample of the word listing index. The word “aboard” was found twice in three documents each.

2.2 Project Layout

- `invert.rb` - the main file that you run.
- `search_engine.rb` - this is where my objects are defined
  - `class SearchEngine` - the main object
    - `initialize` - constructor
    - `print_word_list` - produces a textual output of the first n wordlist entries for the benefit of the grader
    - `query` - takes a single word and produces a query
    - `query_multiple_words` - takes an array of words and produces a query
    - `add` - adds words to the wordlist along with which document they came from and where in the document they were seen
    - `word_does_not_belong` - boolean method to determine if the word should go into the wordlist (i.e. should not be null, empty, or in the stoplist)
    - `df` - document frequency
    - `idf` - inverse document frequency
    - `tf` - term frequency
    - `w` - log frequency weight
    - `tfidf` - term frequency inverse document frequency
    - `get_word_in_context` - given a word and a document, this gets a little snippet of the word with some surrounding context
    - `find_documents_with_all_words` - given a bunch of words, this finds all of the documents that all of the words are in
  - `class String` - adding some new functionality to the base `String` class
    - `split_with_index` - given a string, we split it into a 2D array where each element is a (token, index) pair, e.g. "Hello world" becomes `[[‘Hello’, 0], [‘world’, 6]]`
    - `stem!` - very basic stemming, e.g. strip out garbage and non-alphanumerics and lowercase everything

- `search_engine.test.rb` - unit tests. May be run like `ruby search_engine.test.rb` to run all tests or `ruby search_engine.test.rb --name <test_name_of_test>` to run a single test
- `report` - folder containing documents used to generate this report
- `test` - the corpus data provided by the instructor from Reuters
- `word_list.txt` - output of word list

3 Instructions for Running

This implementation was developed and tested on a Macbook running Mac OS X 10.6.4 running ruby 1.8.7 (2009-06-12 patchlevel 174) [universal-darwin10.0], but no special libraries were used. Therefore this implementation should be compatible with most installations of Ruby.

To run the search engine, simply type `ruby invert.rb` from the local directory where `invert.rb` is located. At this point, the search engine should be waiting for queries. Notice in the example below, we typed `hello` and we see a query results telling us that it was found once in `file03.txt`. Then we type `foo` and we're told that this term was not found at all. Next we type `nasa hello` to test searching on multiple terms. There was only one document where both of those terms appeared: `file03.txt`. The query `nasa project`, on the other hand, yields three documents. If we simply hit enter, we're given instructions to enter a query or type `ZZZ`. If we type `ZZZ` this tells the search engine that we're done and the program ends.

```
$ ruby invert.rb
hello
idf = 1.477
Appeared in ./test/file03.txt 1 times at these location(s): 577
tf = 1, tf.idf = 1.477
  ... ional Space Station Hello World My name i ...

foo
Query returned to results.
nasa hello
./test/file03.txt 2.098
nasa project
./test/file10.txt 1.195
./test/file01.txt 1.195
./test/file03.txt 1.195

Enter a query or type 'ZZZ' to end.
ZZZ
$
```

Listing 2: Sample interaction with the search engine.

4 References

- Formulas were used as per definition from the class website: `http://www.infosci.cornell.edu/Courses/info4300/2010fa/index.html`
- I collaborated with Karan Kurani for this project.