Homework #8: Information Retrieval (IR)

Deadline: 11:59pm Saturday 3 May 2014

Please present a report with all your answers, explanations, and sample images or plots. Submit also a soft copy of the source code and binaries used to generate these results. Please note that copying of any results or source code will result in ZERO credit for the whole homework.

Acknowledgment: This homework is adapted from Chris Manning and Dan Jurafsky's Coursera NLP class from 2012.

In this assignment you will be improving upon a rather poorly-made information retrieval system. You will build an inverted index to quickly retrieve documents that match queries and then make it even better by using term-frequency inverse-document-frequency weighting and cosine similarity to compare queries to your data set. Your IR system will be evaluated for accuracy on the correct documents retrieved for different queries and the correctly computed tf-idf values and cosine similarities.

Data

You will be using your IR system to find relevant documents among a collection of 60 short stories by the famed Rider Haggard. The training data is located in the `data/` directory under the subdirectory `RiderHaggard/`. Within this directory you will see yet another directory `raw/`. This contains the raw text files of 60 different short stories written by Rider Haggard. You will also see in the `data/` directory the files `queries.txt` and `solutions.txt`. We have provided these to you as a set of development queries and their expected answers to use as you begin implementing your IR system.

Your Assignment

Improve upon the IR system given. This involves implementing:

- **Inverted Index:** Implement an inverted index - a mapping from words to the documents in which they occur.
- **Boolean Retrieval:** Implement a Boolean retrieval system, in which you return the list of documents that contain all words in a query (yes, we only support conjunctions...)
- **TF-IDF:** Compute and store the term-frequency inverse-document- frequency value for every word-document co-occurrence:

$$w_{t,d} = (1 + \log_{10} tf_{t,d}) \times \log_{10}(N / df_t)$$

- **Cosine Similarity:** Implement cosine similarity in order to improve upon the ranked retrieval system, which currently retrieves documents based upon the Jaccard coefficient between the

query and each document. The lectures will be helpful for this task. **Also** note that when computing $w_{t,q}$ (i.e. the weight for the word w in the query) do not include the idf term. That is $w_{t,q} = 1 + \log_{10} tf_{t,q}$

Note that the reference solution uses ltc.lnn weighting for computing cosine scores (see the lectures).

To improve upon the information retrieval system, you must implement and/or improve upon the following functions:

- `index()`: This is where you will build the inverted index. The documents will have already been read in for you at this point, so you will want to look at some of the instance variables in the class (analogous data structures are available in the Python code):
  - titles: list of titles
  - documents: list of documents (lists of strings)
  - vocab: list of strings
- `get_posting(word)`: This function returns a list of integers (document IDs) that identifies the documents in which the word is found. This is basically just an API into your inverted index, but you must implement it in order to be evaluated fully.
- `boolean_retrieve(query)`: This function performs Boolean retrieval, returning a list of document IDs corresponding to the documents in which all the words in query occur.
- `compute_tfidf()`: This function computes and stores the tf-idf values for words and documents. For this you will probably want to be aware of the class variables vocab and documents which hold, respectively, the list of all unique words and the list of documents, where each document is a list of words.
- `get_tfidf(word, doc)`**:** You must implement this function to return the tf-idf weight for a particular word and document ID.
- `rank_retrieve(query)`: This function returns a list of the top ranked documents for a given query. Right now it ranks documents according to their Jaccard similarity with the query, but you will replace this method of ranking with a ranking using the cosine similarity between the documents and query.

We suggest you work on them in that order, because some of them build on each other. It also gets a bit more complex as you work down this list.

Evaluation

Your IR system will be evaluated on a development set of queries. The queries are encoded in the file **queries.txt** and are:

- separation of church and state
- priestess ritual sacrifice
- demon versus man
- african marriage queen
- zulu king

We test your system based on the **four** parts mentioned above: the inverted index, boolean retrieval,

computing the correct tf-idf values, and implementing cosine similarity using the tf-idf values.

Running the code

```
$ cd python
$ python IRSystem.py
```

This will run you IR system and test it against the development set of queries. If you want to run your IR system on other queries, you can do so by replacing the last line above with

```
$ cd python
$ python IRSystem.py "My very own query"
```

where **My very own query** is your query.

Note that the first time you run this, it will create a directory named `stemmed/` in `../data/RiderHaggard/`. This is meant to be a simple cache for the raw text documents. Later runs will be much faster after the first run. *However*, this means that if something happens during this first run and it does not get through processing all the documents, you may be left with an incomplete set of documents in `../data/RiderHaggard/stemmed/`. If this happens, simply remove the `stemmed/` directory and re-run!

Requirements

You are required to implement the inverted index, boolean retrieval, correct tf-idf scores, and ranked retrieval with cosine similarity using tf-idf values.

Please submit your code and report in one zip file, named **CMP462.HW03.firstname.lastname.zip**. For example, if your name is Mohamed Aly, your file should be named **CMP462.HW03.Mohamed.Aly.zip**.

Grading

1 pts: report and submission file name
2 pts: correct implementation of inverted index
2 pts: correct implementation of boolean retrieval
2 pts: correct implementation of tf-idf scores
3 pts: correct implementation of ranked retrieval with cosine similarity and tf-idf scores