Data Mining

Project 1

Due: 15/11/2015, 23:59.

Instructions

You must hand in the homeworks electronically and before the due date and time.

Handing in: You must hand in the homeworks by the due date and time by an email to `aris@dis.uniroma1.it` that will contain as attachment (not links!) a .zip or .tar.gz file with all your answers and subject
`[Algorithmic Methods for Data Mining] Project 1`
After you submit, you will receive an acknowledgement email that your project has been received and at what date and time. If you have not received an acknowledgement email within 2 days after you submit then contact the instructor.

The solutions for the theoretical exercises must contain your answers either typed up or hand written clearly and scanned.

The solutions for the programming assignments must contain the source code, instructions to run it, the index that you created, and some sample queries (screenshots).

For information about collaboration, and about being late check the web page.

The project consists in building a simple search engine for news. It will be made of 3 programs called *Collect*, *Index*, and *Search*.

The *Collect* program downloads documents from `http://kijiji.com`. We are interested in the announcements (top announcement and regular announcements but not sponsored ones) for houses for sale in Rome. You need to:

1. Download the web pages with the results from `http://kijiji.com`
2. Parse the result pages and for each announcement extract the *title*, *location*, *price*, description (the short description that appears in the result page), and the url of the full ad. You can use a package such as `BeautifulSoup` to parse the web pages.
3. Store the ads under a directory `documents/`, one ad per file, and numbered `document-000001` to `document-nnnnnn`. Note that this will create too many files, so create subdirectories `documents/documents-000001-000500/`, `documents/documents-000501-001000/`, and so on, where each subdirectory contains the corresponding files.
4. Each file should be in a tab-separated-value (TSV) format, and contain

   ```
   title <TAB> location <TAB> price <TAB> ad URL <TAB> description
   ```

It is very important to wait for a 1–2 seconds between you download two pages, otherwise `kijiji.com` may block you.

The *Index* program builds an inverted index over the downloaded documents. Documents should be pre-processed to remove stopwords, to normalize the vocabulary, and stem (you can use the `nltk` library). The index should be stored in a directory `index/` in two files: `vocabulary.txt` and `postings.txt`. The file `vocabulary.txt` will contain the vocabulary as a TSV file with one term per line:

`termnumber <TAB> term`

The file `postings.txt` will contain the posting lists as a tab-separated file with one term per line:

`termnumber <TAB> document1 <TAB> document2 <TAB> ...`

These are the document ids of the documents (ads) that containg the term corresponding to `termnumber` in their title, location, and description.

The *Search* program receives as an argument a set of words (e.g. prati balcone appartamento). It should read the index in memory. You need to create in memory:

1. The posting lists of the various terms, read from `postings.txt`
2. A map of terms to posting lists, read from `vocabulary.txt`

Return the documents that contain all the terms. Print the title, the location, the price, and the URL. Of course, the query terms should go through the same preprocessing as the ads.
Note: You may compute the result set by computing the intersection using existing python tools, yet for a full score you should implement an efficient way for computing the intersectionn.

For additional bonus points:

- The search could return the results clustered into groups of similar documents.
- The search could have a nice web-based interface.
- You can download the entire adcription and additional inormation stpred in the ad page.
- The *Collect* program could download from more than one ads site.