Programming Assignment for HW3

Homework 3 (Programming Assignment A)

Download data files bundled as a .zip file from hw3data.zip

Each file in this archive contains entries that look like:

journals/cl/SantoNR90:::Michele Di Santo::Libero Nigro::Wilma Russo:::Programmer-Defined Control Abstractions in Modula-2.

that represent bibliographic information about publications, formatted as follows:

paper-id:::author1::author2::…. ::authorN:::title

Your task is to compute how many times every term occurs across titles, for *each* author.

For example, the author Alberto Pettorossi the following terms occur in titles with the indicated cumulative frequencies (across all his papers): program:3, transformation:2, transforming:2, using:2, programs:2, and logic:2.

Remember that an author might have written multiple papers, which might be listed in multiple files. Further notice that 'terms' must exclude common stop-words, such as prepositions etc. For the purpose of this assignment, the stop-words that need to be omitted are listed in the script stopwords.py. In addition, single letter words, such as "a" can be ignored; also hyphens can be ignored (i.e. deleted). Lastly, periods, commas, etc. need to be ignored; in other words, only alphabets and numbers can be part of a title term: Thus, "program" and "program." should both be counted as the term 'program', and "map-reduce" should be taken as 'map reduce'. Note: You do *not* need to do stemming, i.e. "algorithm" and "algorithms" can be treated as separate terms.

The assignment is to write a parallel map-reduce program for the above task using either octo.py, or mincemeat.py, each of which is a lightweight map-reduce implementation written in Python.

These are available from http://code.google.com/p/octopy/ and mincemeat.py-zipfile respectively.

I strongly recommend mincemeat.py which is much faster than Octo,py even though the latter was covered first in the lecture video as an example. Both are very similar.
Once you have computed the output, i.e. the terms-frequencies per author, go attempt Homework 3 where you will be asked questions that can be simply answered using your computed output, such as the top terms that occur for some particular author.

Note: There is no need to submit the code; I assume you will experiment using octo.py to learn how to program using map-reduce. Of course, you can always write a serial program for the task at hand, but then you won't learn anything about map-reduce.

Lastly, please note that octo.py is a rather inefficient implementation of map-reduce. Some of you might want to delve into the code to figure out exactly why. At the same time, this inefficiency is likely to amplify any errors you make in formulating the map and reduce functions for the task at hand. So if your code starts taking too long, say more than an hour to run, there is probably something wrong.