Data Mining

Homework 2

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 the instructor that will contain as attachment (not links!) a .zip or .tar.gz file with all your answers and subject
`[Data Mining class] Homework 2`
After you submit, you will receive an acknowledgement email that your homework 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, and the output generated (to the screen or to files).

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

Most of the questions are not very hard but require time and thought. **You are advised to start as early as possible, to work in groups, and to ask the instructor in case of questions.**

Problem 1. In this problem we will see how we can download data using an API, and practice some of the text processing steps and MapReduce. The goal is to create an inverted index for the abstracts of some articles from the New York Times newspaper.

The NY Times API is available at `http://developer.nytimes.com`. It provides access to various articles, both historic and new. For this assignment we are interested in the *Times Newswire API*, which provides access to articles, blogs, and so on, as they are being produced, and for each article we can obtain directly from the API its URL and its abstract, among other information. Your tasks for this problem are the following:

1. Study the API, download 30,000 *different* articles from the Times Newswire API, and for each of them save the URL and the abstract. Note that because articles are created continually, you may end up downloading some articles multiple times; you should make sure that you do not store each article more than once. For each article assign a unique docID. Note that this part is not completely trivial because, among other issues, you need to deal with the fact that often the API does not return the expected document, so you need to catch the exceptions thrown, put the right delays, and retry, for some steps.

2. Perform some preprocessing of the article abstracts. In particular, remove all punctuation and numbers, and keep only the words. Convert each word to lowercase, remove stopwords, and stem each word using Porter's stemmer. For this part you may find useful the `NLTK` Python package.

3. After preprocessing the articles, build an inverted index using Hadoop. (This is a small dataset and we could build the inverted index in memory but we want to practice.) The final

outcome should be a file in which each line is of the form:

$$\texttt{term}_\texttt{j}\texttt{:doc}_{\texttt{1j}}\texttt{,tfidf}_{\texttt{1j}}␣\texttt{doc}_{\texttt{2j}}\texttt{,tfidf}_{\texttt{2j}}\ldots$$

where ␣ is just a space character, $\texttt{term}_\texttt{j}$ is the jth term (alphabetically), $\texttt{doc}_\texttt{ij}$ is the ith article in the posting list of $\texttt{term}_\texttt{j}$ (sorted by docID), and $\texttt{tfidf}_\texttt{ij}$ is the TFIDF score of $\texttt{term}_\texttt{j}$ in article $\texttt{doc}_\texttt{ij}$. You can process the file before and after the *inversion*, for instance, to put each line in the desired format, but the actual work in which the algorithm calculates frequencies and TFIDF scores and builds the inverted index should be done in Hadoop. Note that one map–reduce round is enough.

Problem 2. Here we will practice developing algorithms for Hadoop by solving the following problem on social networks: Christakis and Fowler, in their book *Connected*, claim that by analyzing some data they have found that we are being influenced in our behavior not only by our friends but also from our friends' friends and even from the friends of our friends' friends, that is, up to disctance 3. Here, given a social network we will design an algorithm to find who are the people who can influence us, according to Christakis and Fowler.

Let us assume that given a directed graph $G = (V,E)$, an edge $(v,u) \in E$ means that v is following u, or that v has declared u being his friend, therefore u has the potential to influence v (but not necessarily the other way around). Now, given a network $G = (V,E)$ and a node $v \in V$, we want to find all the nodes u such that there exists a path from v to u of length at most 3. Design and implement an algorithm for Hadoop that finds all such nodes. Assume that the input file(s) give the adjacency list for each node, that is, they are of the form

$$\texttt{v:u}_1\texttt{,u}_2\texttt{,u}_3\ldots$$

where `v` is a node and each $\texttt{u}_\texttt{i}$ corresponds to the edge $(\texttt{v},\texttt{u}_\texttt{i})$.

First try your algorithm with the simple file `epinions.txt` available at
`https://snap.stanford.edu/data/soc-Epinions1.html`.
After you verify that your program is running, try with a large dataset with more that 100 million users and more than 2.5 billion links, available at
`http://archive.org/details/friendster-dataset-201107`.
(Note that the files are also available as a torrent.)

Problem 3. The k-means clustering problem takes as input n points X in a d-dimensional space and asks for a partition of the points into k parts $C_1, \ldots, C_k$. Each part C_i is represented by a d-dimensional representative point r_i such that

$$\sum_{i=1}^{k} \sum_{x \in X, x \in C_i} d(x,r_i)$$

is minimized. In the k-means problem, the distance $d(x,r_i)$ from a point to its corresponding representative is $d(x,r_i) = \|x - r_i\|^2$, and r_i is the *mean* of the points in cluster C_i.

In class, we mentioned that the k-means clustering problem is NP-hard for $d \geq 2$. However, the k-means problem for $d = 1$ can be solved optimally in polynomial time.

- Design a polynomial-time algorithm for the k-means problem for $d = 1$. Compute the running time of your algorithm. (Hint: Can you solve the problem for k clusters, if you know the solution for $k - 1$ clusters?)

- Use your algorithm to cluster the 1-dimensional data points available at `http://cs-people.bu.edu/evimaria/Italy-2015/SmallDataset-1D.txt` and `http://cs-people.bu.edu/evimaria/Italy-2015/LargeDataset-1D.txt`. You need to decide the number of clusters k for each dataset yourselves; you have to justify your choice of k. If you see that your solution does not terminate, try to think how you could improve its running time.