Data Mining

Homework 2

Due: 7/4/2013, 23:59.

You must hand in the homeworks electronically and before the due date and time. Check the web page for instructions about collaboration, about being late, and about handing in the homework.

Part of the homeworks is dealing with the practical issues that arise, for example, problems when installing Hadoop or a package, or how to download and clean some data. This homework requires the installation of various pieces of software (Java from Oracle, Hadoop, Dumbo, etc.), so you will probably have to spend some time resolving all the problems that arise.

Problem 1. Here we ask you to implement Problem 4 of Homework 1, in Hadoop. In particular, write a MapReduce program, executable in Hadoop, to find the top-10 beers with the highest average overall score among the beers that have had at least 100 reviews. Note two points:

- You may preprocess the file so that you have one line per beer so that afterwards you can process it easily with MapReduce.
- You may need more than one rounds of MapReduce to count and to obtain the top-10 beers.

Problem 2. 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.

Problem 3. 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

`http://aris.me/contents/teaching/data-mining-2013/homeworks/datasets/epinions.txt`.

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`.