

Web Search, Télécom ParisTech

PageRank

Pierre Senellart (pierre.senellart@telecom-paristech.fr)

18 March 2009

The purpose of this lab session is to implement the iterative computation of the PageRank algorithm, to compute the importance of nodes in a graph.

Dataset

The graph of the Simple English Wikipedia is provided as dataset. You can find it either on the course website, or as `~senellart/inf396/labels` and `~senellart/inf396/edge_list` on the computers used during the labs. The set of articles is slightly different from the one used yesterday, since `Category`: articles are kept. The format of the two files is as follows: `labels` contains the title of all articles, one per line, sorted by lexicographical order. Article appearing on line n is implicitly given the index $n - 1$. The first line of `edge_list` is the total number of articles (that is, the number of lines of the `label` file). Each line that follows has this form:

```
A B1,C1 B2,C2 ... Bn,Cn
```

A is the index of a given article (the `edge_list` file is sorted by such indices). B_1, \dots, B_n are all articles pointed to by A and C_1, \dots, C_n the number of links from A to the respective article.

1 PageRank

1. Create a class `MemoryGraph` that will store in memory the content of a (weighted) graph, as an adjacency list for each node of the graph. Implement a constructor that takes as arguments two files in the format described above.
2. Implement the PageRank iterative algorithm on such a graph. Do not forget to normalize the adjacency lists so that the sum of all outgoing edges of a given node is 1. You can use a damping factor of 0.85 and stop the iterative computation when the relative difference between two vectors is less than 1%.
3. Test your implementation. What are the nodes of the Simple English graph that are the most important? Does that sound reasonable?

2 To go further

1. Combine this with the inverted index you built yesterday for the Simple English dataset, so that queries over this dataset use a combination of tf-idf and PageRank. Test it.
2. Combine this with your Web crawler to compute the importance of crawled Web pages in the subgraph of the Web that you retrieved.

3. It is unreasonable to assume that the whole graph can be kept in memory. Look at how you could use the `FileChannel` class of the `text.nio.channels` package to directly work on a disk-based graph. Propose a way to store this graph.