Lab 2 – Inverted index & PageRank on the Wikipedia corpus

Due date: October 16th at 23:59
Status check: October 2nd

Dataset information

Wikipedia offers a compressed XML file for download that contains every article on the site, including redirect and disambiguation pages. We have downloaded the file, decompressed it, and split it into 210 files. The size of each file is between 20MB and 45MB. Each of them contains several thousand Wikipedia articles.
Articles are separated by a pair of tags as in <page>contents</page>. The title of each article is marked by "<title>Name of the article</title>".
Links to other Wikipedia articles are of the form "[[Name of other article]]".
Get the data files at ftp://143.248.133.237 (id & password will be announced in class.)
You can get original XML file at http://download.wikimedia.org/enwiki/20080724/enwiki-20080724-pages-meta-current.xml.bz2

1. **Build an inverted index.**
   Inverted index is a mapping from words in a document to the document identifier. Document identifiers should be hash table values from the titles of each article. Build an inverted index of words to the documents which contain the words. The result will be in the form of (word, docID[ ]).

   Your program should not consider case-sensitivity ('Exam' is the same as 'exam', even 'eXAm') and we recommend to skip punctuations, articles (a/an/the), prepositions, of all kinds of tenses of 'be' verbs (as Wordcount 2.0 did in Lab 1). But ensure that contractions don't change meanings (It's & its have different meaning)

2. **Implement PageRank**
   Turn the given corpus into a graph, calculate PageRank of the graph, and return the PageRank of all the articles in a human-readable form.

One Possible Outline

1. Build a graph - Takes the Wikipedia data and constructs a link graph.
   (k1 = Wikipedia file name, v1 = file contents, k2 = document id, v2 = neighbor id)
2. Calculate PageRank - Iteratively updates the page rank values of a graph.

3. Output the page ranks and article titles in text.

3. Build a simple search engine

Today's search engines use a mixture of complicate algorithms for information retrieval. PageRank is one of them. Here we only use the PageRank algorithm to build a rudimentary prototype of a search engine.

The search engine should run as follows:

1. Take a keyword (only one word)
2. Find all articles that contain the keyword by using the inverted index on a **single VM**.
3. Sort those articles according to PageRank scores.
4. Display titles of top 10 articles according to the PageRank scores.

Turn in

- Source code for inverted index, PageRank, and query program
- Program output:
  - Inverted index: only for the following words (TBA)
  - PageRank: List top 100 pages (titles) with their PageRank scores.
- URL of the search engine
- Report

1. Describe your implementation
2. Explain how to execute your code.
3. Feedback
   - What you've done & what you haven't done.
   - How long you expected this project would take to finish and how much time you actually spent on this project?
   - Acknowledge any assistance you received from anyone and any online resource.

[References]

1. http://hadoop.apache.org/core/docs/current/api/ This URL explains Java classes that are specifically designed for MapReduce on Hadoop. Using file formats in this page should help your implementation.