Web Data Management, assignment 2

Hans van den Bogert 1307983

Bastiaan van Graafeiland 1399101

June 30, 2013

1 Introduction

This document describes the choices made for assignment 2 which in our case was chapter 19 from: "Web Data Management" by Serge Abiteboul et al.

2 Exercise 19.5.1

For this exercise we basically copied the source-code from the example (as was implied by the exercise) and added the `Authors.CountReducer` class as the combiner. This works fine because the counting of author names is associative and commutative.

3 Exercise 19.5.2

Normally hadoop by default works on a granularity of lines for the mappers. In the case of XML-data this does not work. Fortunately a lot of other people have already solved this by using Hadoops extensible `InputFormat`. In particular we used the class `XMLInputFormat` from the *Cloud9* Hadoop toolkit, which only needed minor alterations to work in our setting. Now by using this input formatter, the mappers work on the granularity of enclosing `movie` tags.

4 Exercise 19.5.3

For each query, we created a separate Pig script file. These can be found in `src/main/pig`.

5 Exercise 19.5.4

The inverted file is built using a single MapReduce job: `InverseDocument.java`. The input needs to be a collection of plain text files representing the documents. The output is then in the format of `term (tab) total_tf:(number); idf:(idf value); filename1:(number of occurrences in filename1); filename2: ....`

For building the index, no stop words are used (every word counts). In addition, we convert each word to lower case and strip it of any punctuation or quotes.

We also used a separate Combiner for this job, because the output of the Reducer is different than that of the Mapper.

We use the raw term frequency per document, and the default calculation of idf:

$$\mathrm{idf}(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|}$$