COSI 129a - Introduction to Big Data Analysis (2014Fall)
Assignment2

by
Group 11: Jinfeng Lin, Leifeng Zhou, Yalin Liu, Jing Zou

Brandeis University
Department of Computer Science

Waltham, Massachusetts
October 21, 2014

1.Implementation

We divide this project into two MapReduce tasks. The first mapreduce task is responsible for training and the second part is to read the output of the first mapreduce task to keep on calculating the possibility of the people's profession.

The input of TrainMaper is the article_index of PA1. it produce <profession, <lemma,count>*> pair to feed to reducers. Reducer will count process the pair with same key--profession to calculate the priority P(Y) and add up the words frequency in each profession.

Here is a small trick in calculating the P(Y). Although P(Y) is required to be divided the total number of profession number, the global variable is hard to transfer from mapper to reducer, neglecting the denominator actually do not affect the result, so using the count of each profession as P(Y) also works.

In the test part, we read article_index in mapper and output the <name, <lemma,count>*> pair out to reducer as we do in the former mapper, we also cached the output of TrainMaperReducer. Then compute the P(Y|X) in reducer.

2.Timing

The first task takes quite a short time, nearly 15 minutes while the second task is much much slower. It takes about 6 hour to get the result.

3.Difficulty and What we have learn

In this part of experiments the first thing we have learnt is the limitation of sharing some value between mapper and reducer of hadoop. We may have to use some

other tools like Zoo to control the concurrency and regulate/share the global variable we may need.

One mistake we have ever made is using the local IO methods to read the hdfs file. This mistake lead us to have a deeper understand of the correlation and difference of slave node of hadoop and datanode of hdfs.

We also met some difficult in the implementation. We use text as our input and output, so one task we need to do is reading the incoming text and put them into vector. We firstly use String.replace() method and once read a <lemma,count> we remove it from string thus next time we read only the first pair in the string. We ignored the price of modifying a String, thus the efficiency is extremely low and stuck the whole program. Then we rewrite this part by just reading the String and do not modifying them, finally solved the problem.

4.Not completed

We have finished the requirements listed on document. But some extra works is not done. Like improving the the accuracy of classifier by eliminating the most frequently appeared words and least appeared word. The accuracy is actually not satisfying, even we have 3 chance to guess.