CIS 4930 Introduction to Hadoop and Big Data

Lab Homework 7 Report

1.Creating an Inverted Index

* 1. Project introduction.

In this project, we wrote a MapReduce job that creates an inverted index. That means when we enter a text input with each line containing a line number, a separator, and a value (the text in the line) it outputs an index where each word has a list of locations where the word appears.

* 1. Description on driver: job configuration, input format, output format, etc.
* The driver extends the configured and implements the tool class.
* The input format for the class is KeyValueTextInputFormat
* The fie input path is set args[0]
* The file output path is set to args[1]
* Both output key and value are set to text
* The reducer class is set to IndexReducer
* The mapper class is set to IndexMapper
  1. Description on mapper: input key, value types, output key, value types, and method manipulation on the input key value pairs.
* The mappers input key type is text
* The mappers input value type is text
* The mappers output key type is text
* The mappers output value type is text
* The program uses a Context object to retrieve the file name
* Uses the path.getname() and key.toString() to get the location of the word
* All the values are switched to lower case
* A for loop is used to split the line into words and the context object is used to write the key and the value
  1. Description on reducer: input key, value types, output key, value types, and method manipulation on the input key value pairs.
* The reducers input key type is text
* The reducers input value type is text
* The reducers output key type is text
* The reducers output value type is text
* The reducer has a first value object that checks if it’s the first time the word has been located, if not it adds a comma and uses a context object to write the output key and value
  1. Data flow description starting at the input files.
* The driver sets the input file and the output file as defined by the user
* The data is then sent to the mapper
* The data goes from the mapper to reducer
* The data is then output
  1. Test data and results.
* The test data used was from Shakespeare.tar.gz
* The results were as expected

1. Calculating Word Concurrence
   1. Introduction

The program counts the number of times words appear directly next to each other

* 1. Description on driver: job configuration, input format, output format, etc.
* The driver extends the configured and implements the tool class.
* The input format for the class is set to default
* The fie input path is set args[0]
* The file output path is set to args[1]
* The reducer class is set to SumReducer
* The mapper class is set to WordCoMapper
* The output key class is set to Text
* The output value class is set to Integer
  1. Description on mapper: input key, value types, output key, value types, and method manipulation on the input key value pairs.
* The mappers input key type is LongWritable
* The mappers input value type is text
* The mappers output key type is text
* The mappers output value type is IntWritable
* The mapper uses a for loop and splits the words
* The context object is used to write the key and the value
  1. Description on reducer: input key, value types, output key, value types, and method manipulation on the input key value pairs.
* The reducers input key type is text
* The reducers input value type is IntWritable
* The reducers output key type is text
* The reducers output value type is IntWritable
* The reducer uses a for loop to get the values and adds it to the wordcount for each key
  1. Data flow description starting at the input files.
* The driver sets the input file and the output file as defined by the user
* The data is then sent to the mapper
* The data goes from the mapper to reducer
* The data is then output to the file
  1. Test data and results.
* The test data used was from Shakespeare.tar.gz
* The results were as expected