Benjamin Morrison
Assignment 3 - Lab 7
Oct. 13, 2019
Hadoop

Inverted Index

Introduction

A body of text in the form of Shakespearian plays is used as input for this program to create an index of all words that appear. Using an indexer, the purpose is to produce an index of all the words in the text, and for each word, a list of locations and line numbers. Here is an example output provided in the lab instructions:

honeysuckle 2kinghenryiv@1038,midsummernightsdream@2175,...

The index is to contain an entry for every single word that appears in the input file(s).

Driver

The input file was taken in by the driver where the job was defined, the input arguments were initialized, the input type was specified and the key/value output types were specified. The input was set to KeyValueTextInputFormat in order to correctly ingest the file's text block. Output format is simply Text for outputting to the console.

Mapper

Input key and value types are both Text, which was implemented in the driver. The output key and value types are again, Text. Context is utilized to write these pairs which then passes them to the reducer. Manipulation of these key/value pairs in the mapper consists of getting the file name of the Shakespearian text files in the current directories and appending the filename along with the indices to the StringBuffer.

Reducer

From here, the reducer takes its input from the key/value pairs as Text and outputs them using Context as Text. The values in this case are all being appended to the correct keys, which in this case are the file names along with the line numbers which are being matched to the keys which are the words. The reducer is simply aggregating the values for the same key into one, using a comma as a separator.

Data Flow

The input file was provided as invertedIndexInput.tgz which was decompressed and added to the HDFS. The input file was taken in by the driver where all the parts relevant to the program were defined and initialized. From there, the mapper function takes the input data and produces key, value pairs to be processed. Finally, the reducer takes those pairs and processes them to produce the indices from the input text body.

Results

Below is an excerpt from the large output that this program produces. As you can see, an index of all the words in the input are produced. The word appears on the left, while the right side is the location of the word @ a page number.

```
Clowns        hamlet@48, hamlet@4939,
Clubs         romeoandjuliet@239, kinghenryviii@4479, titusandronicus@837,
Cneius        antonyandcleopatra@3774,
Cnidos        juliuscaesar@47,
```

Word Co-occurance

Introduction

The purpose of this program is to parse through the Shakespearian body of text and produce counts for the number of times that those words appear next to each other. Although the lab is very vague on this, it has been assumed that the goal is to present every possible raw pairing of words that appear in the input that can be observed next to each other and tally those occurrences.

Driver

The input format is by default TextInputFormat, the output format for the key is Text while the output for the value is IntWritable. The job is defined and the input arguments are defined as well here. Output is Text and IntWritable.

Mapper

The mapper takes its inputs from the driver which, defines its inputs as LongWritable for key and Text for value. Its output types are text for key and IntWritable for values. Using StringBuffer, a for loop parses through the text and creates the word pairs which are to be later counted. These pairs are appended with a separating comma in-between them.

Reducer

Finally, the reducer has input as text for the keys and intWritable for the values and remains the same for the output. Here, the input from the mapper is simply counted for each key/value pair passed into the reducer. These new pairs are the output which will be displayed on the console.

Data Flow

Again, the input file was provided as invertedIndexInput.tgz which was decompressed and added to the HDFS. The input is injected by the driver which sets up the job, initializes the input arguments, and initializes the output key, value types and the mapper/reducer types. From here, the input from the shakespearian text is passed into the mapper where the key/value data is created and processed. From here the reducer (sumreducer) takes the key value pairs and finishes processing for output. The occurrences of word pairs is compiled and produced as output, which can be seen in the results section.

Results

Below is an excerpt from the large output that this program produces. The output contains the number of occurrences that the pair of words appears in the input file. The output if formatted to show the word pairs, and the number.

```
and, worse              9
and, worship            2
and, worshipped         1
and, worth              2
and, worthier           1
and, worthiest          1
and, worthiness         1
and, worthless          1
and, worthy             2
and, wot                2
and, would              36
and, wouldst            2
and, wound              2
and, wounding           1
and, wounds             3
and, wrath              4
and, wrathful           2
```