HBASE INVERTED INDEX

Sohil Jain Ashutosh Bhargave

[**sohjain@iu.edu**](mailto:sohjain@iu.edu)[**ashubhar@indiana.edu**](mailto:ashubhar@indiana.edu)

**Goal**

Write an HBase FreqIndexBuilder program to build an inverted index table which has the unique term’s occurrences in all documents from the clueWeb09 dataset. Each row record of columnfamily “frequencies” is unique, where the rowkey is the unique term stored in byte format, column name is the documentId that contains this term, and value is the term frequency shown per document. Note that each row has multiple columns. The result must be loaded to HBase clueWeb09IndexTable. Figure 1 shows the schema of clueWeb09IndexTable.

**Technical Report:**

**Flow:**

Start Hadoop, HBASE and Insert Data files into wordclueWeb09 table in Hbase

Read The Hbase Table in the Mapper and count the frequency of each word in Mapper.

Read The Hbase Table in the Mapper and send the count to the Reducer which calculates overall frequency.

Iterate over all the words in the document and insert those into Hbase keeping document id as the column name

In the Map task we followed these steps,

1] Get key and value as URI and Content.Then, Stored the content in the string named “content”

2] Call “getWordFreq(content)” function which returns the key value pair of word and the count of that word in that particular document,

So, consider I have 100 rows as,

|  |  |
| --- | --- |
| URI | Content |
| Document1 | Indiana University is a good university. |
| Document2 | Indiana is developing state. |
| …. | ….. |
| Document112304 | Indiana University ranks 6th in MIS. |

We will iterate over all these rows and getWordFreq will first give count of all the words in first row as,

Indiana 1

University 2

Is 1

A 1

Good 1

This iterates over all the rows, and we store it in a hashmap.

3] We then iterate over the hashmap and then store the data into Hbase.

For storing data into Hbase, we use the constants file which has the column family name initialized to a value. The column name changes with every document. The column name is nothing but the document id. Here we store the data in to hbase as <word, docID, frequency> The data type is

<ImmutableBytesWritable,Longwritable>.

Put put = new Put(rowKey);

put.add(Constants.CF\_FREQUENCIES\_BYTES, documentID, contentBytes);

**Compile and run your code**

$ cd /root/MoocHomeworks/HBaseInvertedIndexing/

$ ./compileAndExecFreqIndexBuilderClueWeb.sh

**Result**

The result is generated into a project2.txt file. The output is as follows,

scanning table clueWeb09IndexTable on frequencies...

------------0'1------------

00000230265 : 1

------------0'23.08------------

00000235243 : 1

------------0,0.00,1,0.00------------

00000118373 : 1

------------0,0.00,1,0.00,2,0.00------------

00000118369 : 1

00000118370 : 1

00000118371 : 1

00000118372 : 1

------------0,0.00,1,0.00,2,0.00,3,0.00,4,0.00,5,0.00,6,0.00,7,0.00,8,0.00,9,0.00------------

00000118368 : 1

------------0,01euros------------

00000226930 : 1

------------0,1.7,5.0------------

00000231836 : 1

------------0,28804,1690753\_1690758\_1693514,00------------

00000121800 : 1

------------0,4458,360183\_395924,00------------

00000121979 : 1

------------0,5px------------

00000200871 : 4

00000200872 : 4

00000200873 : 4

00000200874 : 4

------------0,8\_------------

00000230251 : 1

------------0,98mb------------

00000108663 : 1

------------0.0,0.0------------

00000110809 : 4

------------0.0,0.0,0.0------------

00000110809 : 4

------------0.0.0------------

00000105847 : 1

00000117450 : 1

00000119432 : 2

------------0.0.0.0------------

00000208104 : 1

00000214044 : 4

00000214055 : 4

00000214058 : 8

------------0.0.1.25------------

00000119025 : 1

------------0.0.2.12c------------

00000209793 : 1

------------0.0.8------------

00000212115 : 3

00000212116 : 3

00000212117 : 3

------------0.0000e------------

00000105277 : 3

------------0.000lux------------

00000124336 : 1

------------0.001g------------

00000122618 : 1

------------0.001lux------------

00000126620 : 1

00000128393 : 1

------------0.001m------------

00000231795 : 1

------------0.002cbm------------

00000122911 : 1

00000122921 : 1

00000124993 : 1

------------0.002lux------------

00000126213 : 1

------------0.002m------------

00000128251 : 1

------------0.002mm------------

00000125347 : 1

------------0.002mv------------

00000124436 : 1

00000124438 : 1

------------0.003mm------------

00000123745 : 1