# Assignment 2

Information Retrieval and Text Mining 20/21

Team Members:

1. Suhas Devendrakeerti Sangolli: (M. No: 3437641)

2. Tushar Rajendra Balihalli: (M. No: 3437638)

3. Stefan Truong: (M. No: 3338287)

## **Task 1**:

### Part A:

- The master splits the documents, then assign these splits to each of the parser machines. Hence, the task description would contain information regarding what split is being assigned to that particular parser, so that it can work on that particular split.

- Upon completion of the task, parser would provide (term, DocID) pairs from that particular split which it received as input from the master. These would further be partitioned based on starting letter of term (ex: a-f, g-p, q-z). These partitions would then be outputted from the parser to the master and the information regarding these files would be present in the task description that parser gives to master. This also serves as information for the master to let master know that this machine has completed its task and is idle again!

- Master assigns idle machines to work as inverter, once the parser completes converting some specified set of splits into partitions. The information regarding these partitions (like address of the file, size, path, etc.) would be present as the task description which master would provide to an inverter, in order to do the reduce operation.

- Upon completing the task, inverter would output the final set of sorted postings lists considering all inputted documents. The information regarding the address, path, etc. of the file would be present in the task description that inverter gives to the master. Then, master can treat this machine as idle and can assign jobs/tasks to this machine later on.

### Part B:

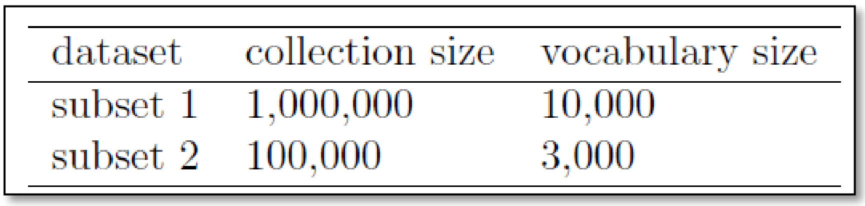
- The number of parsers being assigned depends on the number of document splits present and also the number of machines present. Considering normal number of splits, if we just use one parser, then no parsing jobs run in parallel and we would consume loads of time. Whereas, one parser for each term would operate everything in parallel, but would definitely affect on the cost and maintenance of those many numbers of computers (availability of idle machines). Hence, the solution lies somewhere in the middle, where the master decides on giving parsing tasks to the computers which are idle, only up to specified number of machines (also keep in mind, idle machines are required to do inverter job as well). Once the idle computers are exhausted, then it has to certainly wait for existing parse jobs to complete, then go ahead with next set of jobs.

- Now, coming to the point of partitions, just one partition would mean, only one inverter needs to reduce that particular partition and hence, no parallel operation at reduce end would exist. On the other hand, one partition per term would need plentiful number of inverters, again the problem of availability, cost, maintenance comes into picture. Hence, to be on the safer side, the number of partitions would restrict to a one-digit number (not more than 10), which would give efficient results, considering both the work load, parallelism and availability of machines.

## Task 2:

- Given the scenario, we can make use of Distributed Index (later club it with logarithmic merge), where current updates are initially stored and handled on Master Computer's memory (M0). Let's define a safe machine as Master.
- Master Computer's memory (M0) should be such that it should be capable to hold updates that are updated weekly. It should have capacity at least to store updates for two weeks. This time is considering other machines get free from indexing.
- Once certain limit is reached, it divides large set of updates into blocks.
- Apart from the master, there exist set of machines having different storage capability. Say, 'D1' has capacity greater than 'M0', 'D2' machine has capacity double of 'D1', and so on.
- Master then assigns these blocks to an idle machine having capacity 'D1', which then indexes the updates.
- This block needs two weeks' time for machine to index updated data. Until then, the master treats this machine as busy and searches for other 'D1' machines to assign similar blocks.
- We would need a greater number of 'D1' machines for indexing recent updates.
- After indexing each block, two of such 'D1' machines will transfer the indexed data to the machine having capacity 'D2'. This 'D2' machine will now index the received updated data and would need two weeks of time, then it would combine with another instance of 'D2' machine and transfer its indexed data to 'D3' machine capacity, and so on.
- Thus, we use distributed indexing in initial stages and rather shift towards dynamic indexing as the data grows with updates over weeks, without slowing down the system too much!

## Task 3:



| dataset | collection size | vocabulary size |
|---|---|---|
| subset 1 | 1,000,000 | 10,000 |
| subset 2 | 100,000 | 3,000 |

We know that: $M = k. T^b$

Where, $M \rightarrow$ Vocabulary size, $T \rightarrow$ Collection size, $(k, b) \rightarrow$ Coefficients to be calculated.

Given the following data, we get the two below equations:

$$\mathbf{10000 = k \,.\, (1000000)^b}$$

$$\mathbf{3000 = k \,.\, (100000)^b}$$

### Subtask 3.1:

Apply (log) on both sides of both the equations, then simplify both in order to get the following results:

$$\mathbf{k = 7.37 \,;\, b = 0.522}$$

## Subtask 3.2:

M = (7.37) * $(100000000)^{(0.522)}$

**M = 110526 (approx.)**

## Task 4:

Given number = 216.

→ $(216)_{10} = (1101\ 1000)_2$

Then, the Variable Byte code for the above number becomes:

**(0000 0001 1101 1000)**

Also, the gamma code becomes:

$(216)_{10} = (1101\ 1000)_2$

Offset → 1011000 (leading bit chopped off from binary code)

Length of this offset = 7

Hence, encoding length in unary code becomes: $(11111110)_1$

Therefore, Gamma code being the concatenation of length and offset →

**111111101011000**

## Task 5:

Given Gamma code → 1110110001001100000

We can easily bifurcate between length of offset and the offset itself:

Length of first offset = 4

First offset = 1100

Binary = 11100

Hence, in Decimal, Gap sequence = 28.

Similarly, next gaps = 2, 4.

**Overall Gap sequence → 28, 2, 4.**

**Hence, the postings lists are → 28, 30, 34.**