DIT065 - Computational Techniques for Large-scale Data Assignment 5

Huw Fryer
Cem Mert Dalli
Stefano Ribes

May 10, 2022

1 Problem 1

Please refer to the comments in the attached code in file `problem1.py`. The output of the program analyzing the file `/data/2022-DIT065-DAT470/gutenberg/060/06049.txt` is the following:

- Number of unique words: 30095 (including empty word "")
- Average word length: 4.180031322236737 (assumed non-unique)
- Average prefix length: 6.144827586206897

2 Problem 2

Problem 2(a)

The data are a series of key/value pairs, which appear to be in a CSV format. The code in the lecture performs one map operation and one reduce operation on the data, in order to establish the duplicates which exist.

The big O complexity is linear, in that it goes over the items in the collection only twice, but it is worth noting that N is quite large which impacts even linear computations. In addition, the overhead for map-reduce jobs is significant. In order to share across 16 nodes, data must travel across the network, and file reads are additionally required. As 86% of the items are unique, this is incredibly wasteful since we do not care about them, and makes the second pass N far higher than it should be.

In the event that the duplicates code were to be extended to counting the frequencies of records with incidences of 4 or more, another pass would be required in order to extract the results which were of four or more, as opposed to merely 1-3 incidences.

Problem 2(b)

Sketch a serial solution using Bloom filters to accelerate this counting; please determine k, the number of hash functions, and b the size of Bloom filters. Choose error rates to obtain accurate counts.

There are 1 billion records, and no memory constraints specified, so the size b of the bloom filter and k the amount of hashes will be calculated based on the error rate p. The requirement is that it obtains accurate counts, which leaves some discretion.

We calculate the memory requirement b and hashes k for progressively lower false-positive rates with the equations, with the results displayed in Table 2. 5.58GB, required for 1 in 1 billion rate of false positives is feasible for commodity hardware. Note that these calculations are not used directly in problem 2(c), since they are calculated by the pybloom-live package, which only offers capacity and error rate as arguments.

$$b = \frac{nln(p)}{(ln2)^2} \tag{1}$$

$$k = \frac{-ln(p)}{ln2} \tag{2}$$

Error rate p	Bloom size b	Hashes k
1e-05	2.79 GB	16
1e-07	3.91 GB	23
1e-10	5.58 GB	33

Table 1: Requirements for memory and hashes for a bloom filter of 1 billion records

The algorithm for the bloom filter will follow this pattern:

$b \leftarrow bloomfilter$
$f \leftarrow 4ormore$
$d \leftarrow duplicates$
while records remain **do**
    $r \leftarrow nextrecord$
    if r seen 3 times **then**
        add r to f
        delete r from duplicates
    else if r not in b **then**
        Add r to b
    else
        increment r value in d
    end if
end while
Iterate $4ormore$ to see frequencies $r >= 4$

Problem 2(c)

The code for 2(c) can be found in the file problem2c.py.

3 Problem 3

Problem 3(a)

As frequently outlined in Lecture 7, 8 and 9 by Alexander Schliep, the probability of setting the i^{th} bit after inserting n items is as following:

$$P(bit\, i\, set\, while\, inserting\, n\, items) \equiv 1 - exp(\frac{-kn}{|b|})$$

Using linearity of expectation, the total number of bits set after n items inserted can be calculated through the below-written formula:

$$E(X) = |b|(1 - exp(\frac{-kn}{|b|}))$$

False negatives after deleting an item would occur only if the bits set for deleted item are also used to define other items. If the bits for the deleted item, *i.e.* the $n+1^{\text{th}}$, has no collusion and the $n+1^{\text{th}}$ item has been using empty bits after inserting n items, its removal would not cause any false negative results. Therefore, we will think of a scenario where the $n+1^{\text{th}}$ item is inserted to the empty bits and calculate its probability.

Assuming a bloom filter B with b bits and with $E(X)$ of those bits are set, the number of empty bits will be:

$$M(X) = |b| - E(X) = |b|(exp(\frac{-kn}{|b|}))$$

We can now calculate the probability of using the empty bits when inserting the $n+1$ element into the bloom filter. In other words, if we want to avoid false negatives, we should always pick empty bits when we draw them randomly for k times.

As detailed in Lecture 9, an hypergeometric RV is appropriate when we have a binary outcome and a sampling without replacement. Therefore, the probability of deleting one item from B not causing false negative results (referred as T event), is the following:

$$P(T) = \frac{\binom{M(X)}{k}\binom{E(X)}{k-k}}{\binom{b}{b-k}} = \frac{\binom{M(X)}{k}}{\binom{b}{b-k}}$$

Problem 3(b)

For question 3a, we calculated the probability of **not** causing false negative results, *i.e.* $P(T)$. The probability of causing false negative results is therefore $P(T^C) = 1 - P(T)$, as the entire sample space is 1.

Using the linearity of expectations once again, below we estimate the number of items for which `query()` will return a false negative result after removal of one element:

$$E(A) = |b|(1 - P(T))$$