CT-LSD A7

Albin Jansfelt, Amanda Allgurén, Lukas Martinsson

May 2022

1 Problem 1

When going from rank i to rank $i+1$ we split the bloom filter in to 2 halves and take the OR operation between the two. In our case we have a bloom filter of length $b = 2^r$ with a total of $2n$ bits set. By splitting, we divide the bloom filter into equally sized halves ($|b/2| = |b_1| = |b_2|$) and both halves will have n bits set.

The expected number of items after taking OR will be the expected number of items in one half (n) plus the expected number of items in the other half which will not collide in index with the first. This can also be seen as spreading out $2n$ bits over a bloom filter size of $|b/2|$ and removing the expected number of collisions.

Let X be the expected number of bits set in b_1 OR b_2.

The expected number of collisions will be,

$$E(collisons) = n * \frac{n}{|b_1|}$$

The expected number of bits in b_1 OR b_2 is,

$$E(X) = n + n * (1 - \frac{n}{|b_1|})$$

alternatively expressed as,

$$E(X) = 2n - \frac{n^2}{|b_1|}$$

2 Problem 2

Consider an array A with prefixes of the k first characters. The array consist of pointers to tries, which indicates each prefix and the count of that prefix. Let n be the number of words we want to sample. The first sampling will describe which prefix-trie we want to sample from. This sampling will be based on the frequency of each prefix. For example, if $n = 3$ and $k = 2$ the first sample could yield a list [2,1,1], which means that we want to sample from "ac" once and "ab" twice (if they are ordered alphabetically). We can then aggregate how many times we should sample from each trie to improve efficiency.

The second sampling will traverse down the particular trie according to the frequency of its children. The end of a word is indicated by an end-sign (such as $). When an end-sign is reached, the word reached will be the full word to sample.

Each trie stores its own frequency and its children. To sample according to frequency, each child will be sampled with the probability $p = \frac{\text{freq\_of\_child}}{\text{freq\_of\_ parent}}$. A new sample, using for example np.choice, will be done in each new node encountered. For example, if a parent has two children (words), one with count 7 and one with count 3, the probability of sampling the first word is 7/10 and the second 3/10, since 10 is the total count and is stored in parent.

```
'''
attributes in a trie object
'''
Trie:
    children = []
    count = 0
    letter = ''

'''
loop through all words to build the trie
'''
buildSamplingStructure():
    T = Text corpus
    A = []
    for word in T:
        prefix <- word[0:k]
        if prefix not in A:
            trie = new Trie()
            A.add(trie)
        elif:
            index = find index in A where prefix == trie.letter
            trie = A[index]
        suffix <- word[k:end]
        trie.goDownTrie(suffix)

'''
traverse down the particular trie
if full word is found, increment its count by one
else add the word to the last node visited.
'''
goDownTrie(suffix):
    if suffix == "":
        return

    letter <- suffix.getFirst()
    if letter in trie.children:
        trie = child, where letter == child

    elif:
        child = new Trie('letter', 1)
        trie.children.add(child)
        trie = child
    trie.goDownTrie(suffix[1:])

'''
traverse down tree, sample in each node where to go next
if end node '$' encountered, return built word
'''
sample_trie(trie, prefix, n):
    if trie == '$':
        return

    tot_count = trie.count
    p = []
    for child in trie.children:
        prob = child.count / tot_count
        p.add(prob)
    children = sample n children with prob p
    child_agg = aggregate number of times to sample each child
    result = []
    for each child in children:
        n = child_agg[child]
        result.add(sample_trie(child,prefix+child.string, n))
    return result

sample_array(A, n):
    tot_count  = 0
    for a in A:
        tot_count += a.count

    p = []
    for a in A:
        p.add(c/tot_count)

    #store p for future samplings

    result = []
    children = sample n children with prob p from A
```

```
child_agg = aggregate number of times to sample each child
for each child in children:
    n = child_agg[child]
    result.add(sample_trie(child, "", n))
return result
```

3 Problem 3

Let the random variable X be the number of draws from the set $\{1, \ldots, n\}$ until all unique values have occured. Further, let x_i be number of draws until the i:th unique value occurs, given that $i-1$ unique items have already occured. Then $X = x_1 + x_2 + \cdots + x_n$. Moreover, x_i is geometric distributed with probability $p_i = \frac{n-(i-1)}{n}$ and its expectation is $E(x_i) = \frac{1}{p_i} = \frac{n}{n-i+1}$.

The expected number of samples before each unique item has occurred is therefore given by,

$$\begin{aligned} E(X) &= E(x_1 + x_2 + \cdots + x_n) \\ &= E(x_1) + E(x_2) + \cdots + E(x_n) \\ &= \frac{n}{n} + \frac{n}{n-1} + \cdots + \frac{n}{1} \\ &= n \sum_{k=1}^{n} \frac{1}{k} \end{aligned}$$