Minimum number of special characters in a merged string

Jacob L. Fine

Jun 19th, 2024

Suppose we are merging pairs of k-mers of equal length l, where each pair of k-mers may overlap with each other by k characters. And each k-mer contains a subset of characters which we denote ‘special characters’, such that the total number of characters in each k-mer is at least p.

We are specifically interested in finding a lower bound on the relative proportion of special characters in the merged string, accounting for the fact that the special characters may fall entirely in the overlap region k. We will denote this as ℓ_{min}. This quantity may be of interest if we want to ensure that merging k-mers does not cause the proportion of special characters in each k-mer to drastically decrease, and want to restore our confidence in our methods by finding a lower bound.

To start, we consider the length of the merged k-mers L, which is

$$L = 2l - k$$

since each k-mer has length l, and the overlap region is of length k.

We will then define some new variables, let ρ be the number of special characters in either k-mer present in the overlaping region, such that $\rho \leq k$. We will also denote the total number of special characters in each k-mer, $p_1 \geq p$ and $p_2 \geq p$, where the lower bound is p since each k-mer must have at least p special characters. The number of unique special characters in each k-mer, i.e., not present in the overlapping region, is therefore $P_1 = p_1 - \rho$ and $P_2 = p_2 - \rho$. We can now formally define an expression for the proportion of special characters in the merged k-mer, using the above variables, and call it $\ell(\rho, k)$. Therefore

$$\ell(\rho, k, P_1, P_2) = \frac{P_1 + P_2 + \rho}{2l - k}$$

We are interested in finding the smallest possible value of the above fraction, obtained from a combination of variables ρ, k, P_1, P_2.

To solve this, we may consider the boundary condition when there exist no unique special characters in either k-mer, implying that $P_1, P_2 = 0$. This occurs only when all the special characters are found in the overlapping region of length k, so in this case, we have that $k = \rho$. We therefore obtain that

$$\ell(\rho = k, P_1 = 0, P_2 = 0) = \frac{\rho}{2l - \rho}$$

We can also show that this is in fact the global minimum using the following reasoning. First, by substituting $P_1 = p_1 - \rho$ and $P_2 = p_2 - \rho$, we can write that

$$\ell(\rho, k, p_1, p_2) = \frac{p_1 - \rho + p_2 - \rho + \rho}{2l - k}$$
$$\ell(\rho, k, p_1, p_2) = \frac{p_1 - \rho + p_2}{2l - k}$$

Since $p_1, p_2 \geq p$, we have that

$$\ell(\rho, k, p_1, p_2) \geq \frac{2p - \rho}{2l - k}$$

We may now assume that p and l are some constants from our real-world problem, in which case this becomes an optimization problem with two variables. We can therefore define an objective function to minimize:

$$\ell(\rho, k) = \frac{2p - \rho}{2l - k}$$

Our explicit goal is therefore to find the value of ρ and k that minimize ℓ. Here, we can use some computational or numerical method to explore possible values and their effect on the overall objective function. Let us use a grid search, which manually explores all combinations of ρ and k that meet some range criteria.

```r
# Define the objective function
objective_function <- function(rho, k, p, l) {
  return((2 * p - rho) / (2 * l - k))
}

# Define constants for p and l
p <- 8
l <- 15

# Define the grid of values (rho, k) to search with integer steps
grid_rho <- seq(0, p, by = 1)
grid_k <- seq(0, l, by = 1)

# Initialize variables
working_min_value <- Inf  # Initialize with a large value for minimization
argmin_rho <- NULL
```

```r
argmin_k <- NULL

# Loop through the pairs of values in the grid
for (rho in grid_rho) {
  for (k in grid_k) {
    # Here we check if our point satisfies our constraints
    if (rho <= p && rho <= k && k <= l) {
      # Evaluate the objective function at the pair of points
      possible_min <- objective_function(rho, k, p, l)

      # Update the minimum value and corresponding argmins if necessary
      if (possible_min < working_min_value) {
        working_min_value <- possible_min
        argmin_rho <- rho
        argmin_k <- k
      }
    }
  }
}

# Print the result
cat("Minimum value of ell:", working_min_value, "\n")
```

```
## Minimum value of ell: 0.3636364
```

```r
cat("Argmin (rho, k):", argmin_rho, argmin_k, "\n")
```

```
## Argmin (rho, k): 8 8
```

As can be seen, the argmin occurs when $\rho = k = 8$, as was the case here. This is consistent with our analytical solution.