Coding and Algorithms for Memories- Final Project
Pseudo-clustering algorithm

Dganit Hanania-Buchris

February 2022

1 Introduction

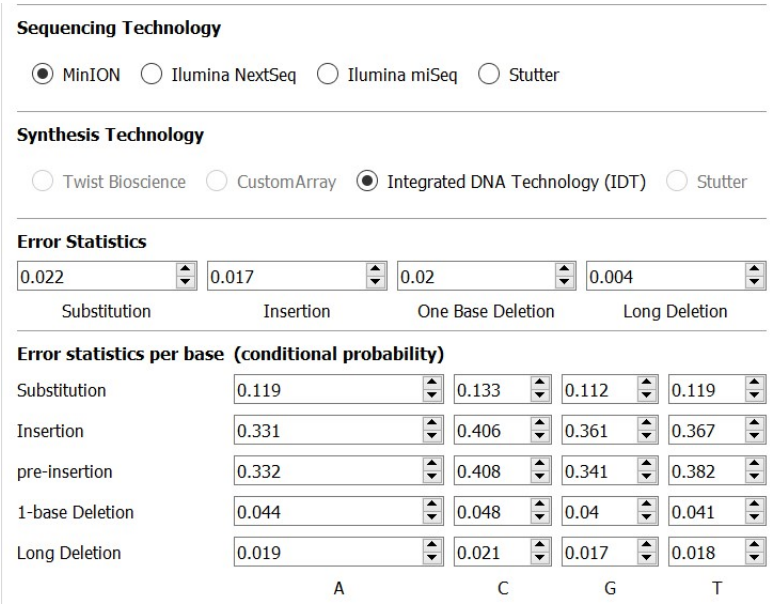
1.1 The Problem

When looking at the DNA storage channel model, there are few levels for decoding the data from the DNA sequences. One of the levels is clustering the DNA strands. The goal of this project is to design and implement a heuristic algorithm for the clustering problem in DNA storage under high error rate from Nanopore sequencing.

The code that I wrote can be found here.

1.2 The Data

To get data to work on, I used a file of 450 DNA sequences. I concatenated to the beginning of each one of the DNA sequences an index.
The indices where taken from a file of about 60 thousands indices with edit distance of three between each two of them. The length of the indices in this file is 11 bases. I took from this file only 450 indices and all of them started with $'AAA'$ so I dropped the beginning of them and made them indices of length 8. The file of this indices is called "450indices.txt".
After I did that, I used the DNA Simulator to generate a file with noisy copies of the original data.
The properties of the erroneous data that the simulator created, including the error statistics are shown below.



There are two output files of the DNA Simulator:

1- A file with all the noisy copies of all of the sequences shuffled. The clustering algorithm will use this file. This file is called "errors_shuffled.txt"
2- A file with a list of all the original sequences and below each one of them, the erroneous copies of it. This file will be used in order to check the success of the clustering algorithm that I will use. This file is called "evyat.txt"

2 Clustering The Data

2.1 Preliminary Work

I created these dictionaries:
1- A dictionary that maps between an index with at most one substitution error to the original index. This dictionary is containing the original indices and the original indices with one substitution error.
2- A dictionary that maps between an index with one deletion error to the original index.
3- A dictionary that maps between an index with one insertion error to the original index.

In all of these three dictionaries, the original indices are the indices from the file "450indices.txt".

4- A dictionary that maps between the erroneous copies and their correct index. I created this dictionary from the file "evyat.txt". This dictionary will help to check the success of the clustering algorithm.

2.2 First Division Into Clusters

In this part, we will go over all the strands' indices.
If the index is in the original indices list, we will assume that the index has no errors and the strand will move to the cluster of this index.
Else, we will check if the index belongs to one of the first three dictionaries that I described above.
In more details, If the strand's first idx_len letters belong to the first dictionary- we will assume that there was one substitution error in the index. So, the strand will move to the cluster of the index after the suitable substitution, according to this dictionary.
Otherwise, if the strand's first $idx\_len + 1$ letters belong to the third dictionary- we will assume that there was one insertion error in the index. So, the strand will move to the cluster of the index before the suitable insertion, according to this dictionary.
Otherwise, if the strand's first $idx\_len - 1$ letters belong to the second dictionary- we will assume that there was one deletion error in the index. So, the strand will move to the cluster of the index before the suitable deletion, according to this dictionary.
The strands that weren't mapped in this level will move to the second level of division.

Two notes:
1- I found that it is better to not include the checking in the second dictionary (the indices with deletion) because it is not adding a lot of mapped strands but it adds a lot of strands that are not mapped correctly. The comparison between these two options will be described in the results part.

2- The indices have edit distance of three between each two of them. It means that in general, we are able to correct one error in the index filed. It means that if one error occurred, we will be able to restore the original index. It is true if the index field is separated from the data. In our case, after deletion or insertion the index's length is changed. The index's field is not separated from the data so deletion or insertion will affect the data field and we will not be able to know if the index's length has changed. This problem is causing that in some cases, some strands' indices will appear in two or more dictionaries from the three first dictionaries described above. Because of this problem, if the strands' index is not in the original indices list, I checked if the strands' index is in more than one dictionary. If it is, I saved this strand in a dictionary that maps between a strand and some indices candidates (the indices from the relevant dictionaries). In the next level, I'll check which of this candidates is better according to a different criterion.

2.3 Second Division Into Clusters

First, I will define $k-mer$ as a sequence of k symbols from $\{A, C, G, T\}$. A $k-mer$ histogram of a strand will contain the amount of each $k-mer$ in the strand.
For example, the $2-mer$ histogram of the strand $ACACGT$, will be as follow: AC:2 CA:1 CG:1 GT:1 and zero for the rest $2-mers$.

In this part, we will go over all the clusters that we created in the first level. For each cluster, I calculated the histogram of 3-mers of $min(clusters'\_size, 10)$ strands from the cluster, and calculated these results mean.
I calculated the histogram of $3-mers$ for each of the strands that weren't mapped at the first level. By using L2 norm, I found for each strands' histogram the closest clusters' histogram and I mapped the strand to this cluster.

Note: According to note 2 in the previous section, for some of the strands I saved some indices candidates. For these strands I compared their histogram only to the specific clusters histogram that are associated with those candidate indices. I did that to save running time.

3 Results

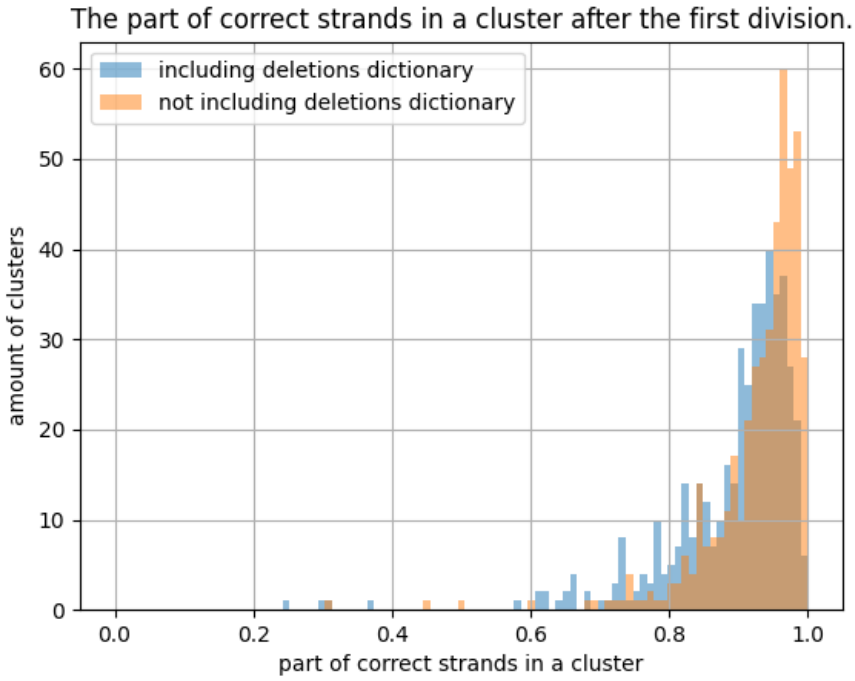
3.1 Results After The First Division Into Clusters

This results are obtained after not including a check of the index in the deletion dictionary. The results after including a check of the index in the deletion dictionary are shown in parentheses.

Total number of strands: 72108.
The percentage of strands that were divided in the first division: 65.55% (compared to 69.15%).

From the divided strands:
The percentage of strands that were divided correctly in the first division: 95.05% (92.4%).
The percentage of strands that were divided falsely in the first division: 4.95% (7.6%).
The minimum percentage of correct strands in a cluster: 31.25% (25%).
The maximum percentage of correct strands in a cluster: 100% (100%).
The mean of the percentage of correct strands in a cluster: 92.93% (89.3%).
The standard deviation of the percentage of correct strands in a cluster: 0.0729 (0.096).
The median of the percentage of correct strands in a cluster: 95.25% (92.31%).

The part of correct strands in a cluster after the first division results are presented in the figure below.



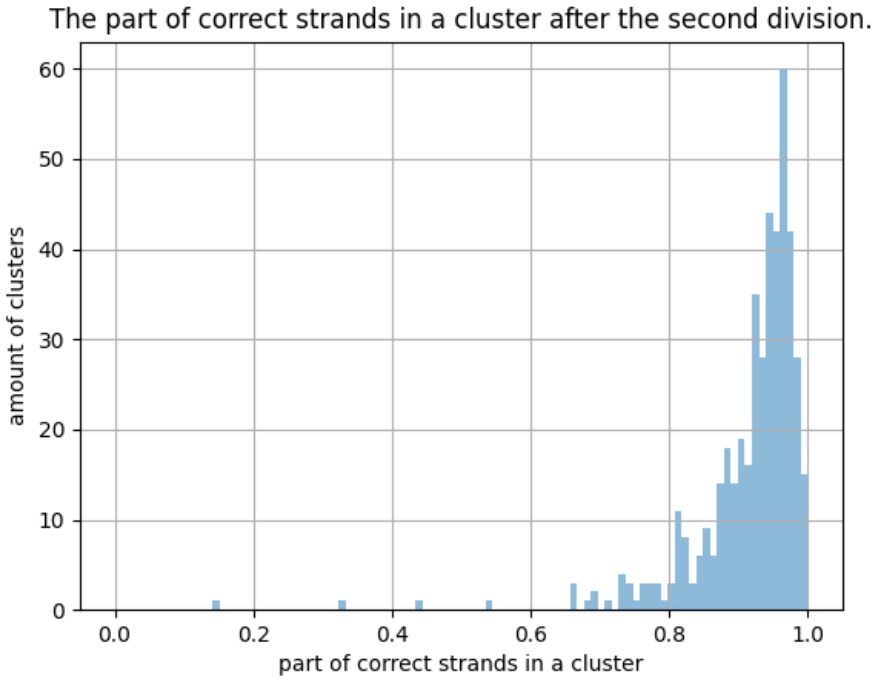
3.2 Results After The Second Division Into Clusters

These results are obtained after treating in a different way to strands that had indices candidates and all the rest strands that were not mapped in the first division.
The results after treating these two kind of strands in the same way (ignoring the candidates), are shown in parentheses. These results are better but the running time of this method is larger.

The percentage of strands that were divided in the second division: 34.45%.
The amount of strands with candidates: 15726.
The amount of strands with no candidates: 9112.

From the strands that were divided in this level:
The percentage of strands that were divided correctly: 92.29% (99.34%).
The percentage of strands that were divided correctly from the strands with candidates: 88.38%.
The percentage of strands that were divided correctly from the strands with no candidates: 99.06%.
The minimum percentage of correct strands in a cluster: 14.29% (31.58%).
The maximum percentage of correct strands in a cluster: 100% (100%).
The mean of the percentage of correct strands in a cluster: 91.61% (94.79%).
The standard deviation of the part of correct strands in a cluster: 0.084 (0.064).
The median of the percentage of correct strands in a cluster: 94.11% (96.48%).

The part of correct strands in a cluster after the second division results are presented in the figure below.



Note: when I dropped at the first level of the division all of the indices that are not in the original indices list, I got better results.
There are some disadvantages in this method:
1- It is much slower. This is because in the first level we divide only about 47% of the strands into clusters.
2- Especially for clusters with few strands, if an error happened in the index field of all of the strands in a cluster, we might loss this cluster.

4 Future Research

There are some optional directions for future research.
One optional direction is, in the second division part, we can use the k-mers histograms as features for machine learning clustering algorithms, such as K-means algorithm.