Progress and Engagement Report

Rohan Hariharan

registration: 100251167

1 Introduction

What are K-mers and why do we want to compare the distnace between them? K-mers are substrings of length ‘k’ of a larger string contained within a biological sequence. A given sequence may be composed of all possible kmers in its alphabet, or by only a subset of kmers that are possible.The concept of kmers is used widely within computational genomics, both for mathematical reasons having to do with their application in graph theory, as well as for practical computational reasons having to do with the way they can be efficiently stored in memory and compared as a way of measuring similarity of genomic regions.
Usually, the term k-mer refers to all of a sequence’s subsequences of length ‘k’, such that the sequence AGAT would have four monomers (A, G, A, and T), three 2-mers (AG, GA, AT), two 3-mers (AGA and GAT) and one 4-mer (AGAT).

We compare K-mer distances because K-mer based distance estimation is an alternative to estimating evolutionary distance based on multiple alignments. At a high level, the distance between two sequences is defined by first collecting the set of k-mers (subsequences of length k) occuring in the two sequences. From these two sets, the evolutionary distance between the two organisms is now defined by measuring how different the two sets are. The more the two sets look alike, the smaller is the evolutionary distance.

2 Aims and Motivation

2.1 Aims

- To generate a random sequence of strings
- To separate the strings into smaller k-mers(length ’k’)
- To compare the k-mers and find distance between them
- Compare Levenshtein and K-mer distances (include scatter plot and correlation graphs)
- What to do if the difference between the k-mer distances is too large

- If time permits, discuss gaps and how to reduce them.
- If time permits, include De-Bruijin graphs

2.2 Motivation

The need for the comparison of strings is highly relevant in modern microbiology. String comparison algorithms are the pathway to determine various characteristics of genomes, DNA or protein sequences. By comparing the sequences of genomes of different organisms, researchers can understand what, at the molecular level, distinguishes different life forms from each other. In general, we can compare two sequences by placing them above each other in rows and comparing them character by character. The main motivation for estimating evolutionary distance based on k-mers, is that it is computationally much faster than first constructing a multiple alignment. Experiments show that phylogenetic tree reconstruction using k-mer based distances can produce results comparable to the slower multiple alignment based methods.

3 Issues and Problems

3.1 Known Issues so far

- The topic comes under Computational Biology which I have very limited experience in.
- Lots of resources that can lead to other tangents.
- There is no cap to the project and can go on to non-important topics that may not be very relevant.

3.2 Addressing Issues

To solve my limited experience of Computational Biology, research is being conducted into DNA strings and k-mers. I have also been multiple papers on the topic of k-mer string comparison.

With regards to the vast amount of resources on the topic that may lead to other tangents, I have decided to use only a couple of research papers as my main sources of

information and the others will be used to fill in the gaps of that the main papers cannot cover.

I have made a MOSCOW analysis(figure 3.2.a) on the topic. I will be using it as a guideline to keep me focused on the work that is necessary and use it to identify what is required and must be done as a baseline for this project and then build on that base.

Due to there not being a defined end point, I will designate a specific area of time to the project and see how far it can go within the time constraints of the project.

4 Design and Planning

This will be split into 2 sections; a build section, and an experimental section to look at other ideas and tangents.

4.1 Build: Algorithm to compare K-mer distances

4.1.1 Implementation

- Language: Python. This is because python would work well for small algorithms with the libraries that it has. Python offers concise and readable code.. The lines of code you have to type for implementing the algorithm goes down drastically in Python.
- Libraries: khmer, numpy, scipy, pandas. numpy, scipy and pandas will be used to create scatter-plot and correlated graphs. khmer can be used to count the k-mers

Determining the possible k-mers of a read can be done by simply cycling over the string length by one and taking out each substring of length ‘k’. The pseudocode to achieve this is as follows 6.2 as 4.1.1.a: Determining k-mers:

4.1.2 Testing

During development the individual components will be unit tested as they are written. After completion, I will ask my Prof. Moulton to review my work to give me an idea of what I can further improve on.

4.2 Experimental: Alternate Distances and measurements

- The section is not fully planned yet as I am still focusing on the distance measurement of k-mer. But once I am done with this, I plan to move onto investigation other distances and how to to measure them.
- Look at suffix trees for q-gram distance profiling.
- Consider looking at De Bruijin graphs and how they can be used

5 Evaluation of progress

At the start of the project I made a Gantt chart(see figure 5.a). This has been updated to track progress over the semester as shown in figure 5.b. As shown in the Gantt chart, progress has been slow and I am behind schedule. I had aimed to start the programming of the algorithms by end of semester 1, but underestimated how long it took me to understand the main premise of the topic. I still aim to start programming during the Christmas break and catch up on the time I lost due to my mental health. I have also decided to work on the basic model of the project first. As of now I am still completely researching k-mers and the methods of comparing the distance between them. And the design for the algorithms is unfinished. I hope to get a majority of this work done by the start of the second semester

6 Appendix

6.1 3.2.a: MoSCoW Analysis

MoSCoW Analysis			
Must	Should	Could	Will Not
• Review K-mers(aka q-grams), Levenshtein, Splitstree • Explain Levenshtein, K-mer, Splitstree • Random Sequence Generator, K-mer Distance Calculator	• Compare Levenshtein and K-mer • Using q-grams, use split networks • What to do if the difference in k-mers is too big • Gaps and how they can be removed	• Scatterplot or Co-Related graphs to compare the distances • Compare to Hamming Distance • Other distances and methods of calculating them • Suffix Trees • De Bruijin Graph	• Detailed Alignment Algorithms

6.2

Algorithm 1 : 4.1.1.a: Determing K-mers

function $(k)mers\ (string\ seq, integer\ k)$:

REQUIRE: k

REQUIRE: arr

$L \leftarrow length(seq)$

$arr \leftarrow new\ array\ of\ L - k + 1\ empty\ strings$ ▷ *iterate over the number of k-mers in seq,* ▷ *storing the nth k-mer in the output array*

FOR $n \leftarrow 0$ TO $L - k + 1$ *exclusive* **DO**

$arr[n] \leftarrow subsequence\ of\ seq\ from\ letter\ n$ *inclusive to letter* $n + k$ *exclusive*

RETURN arr

6.3 5.a: Original Gantt Chart

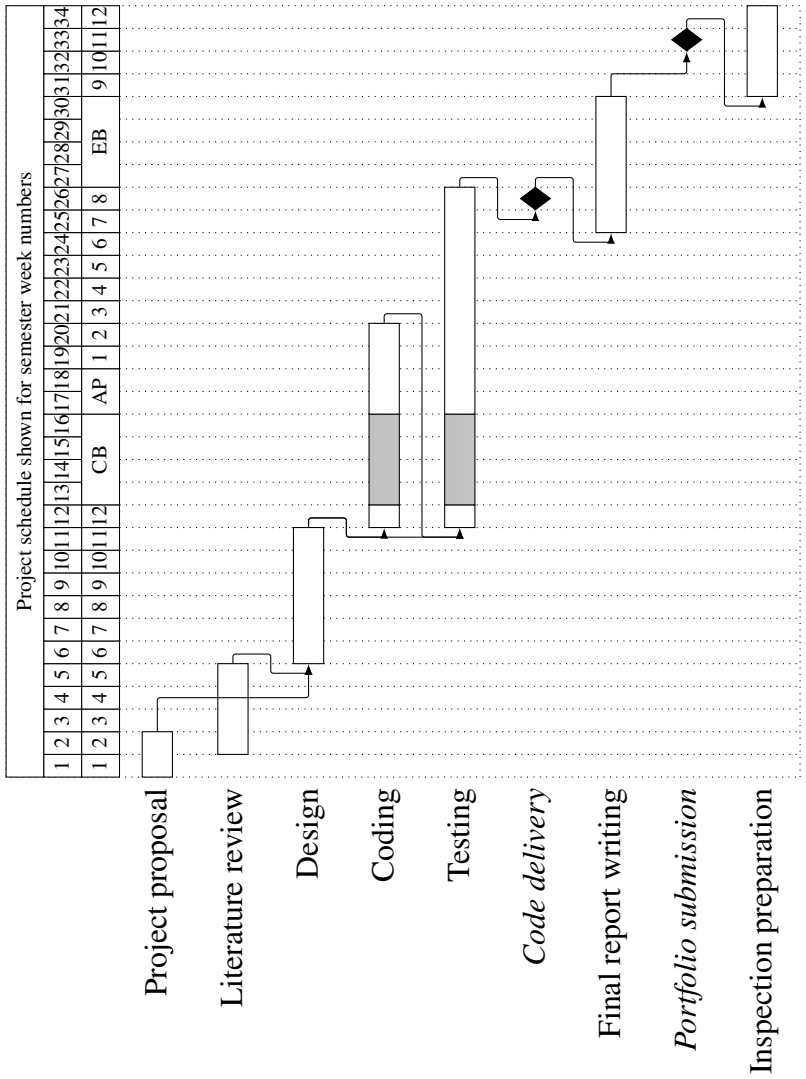

Figure 1: 5.a Original Gantt Chart

6.4 5.b: Modified Gantt Chart

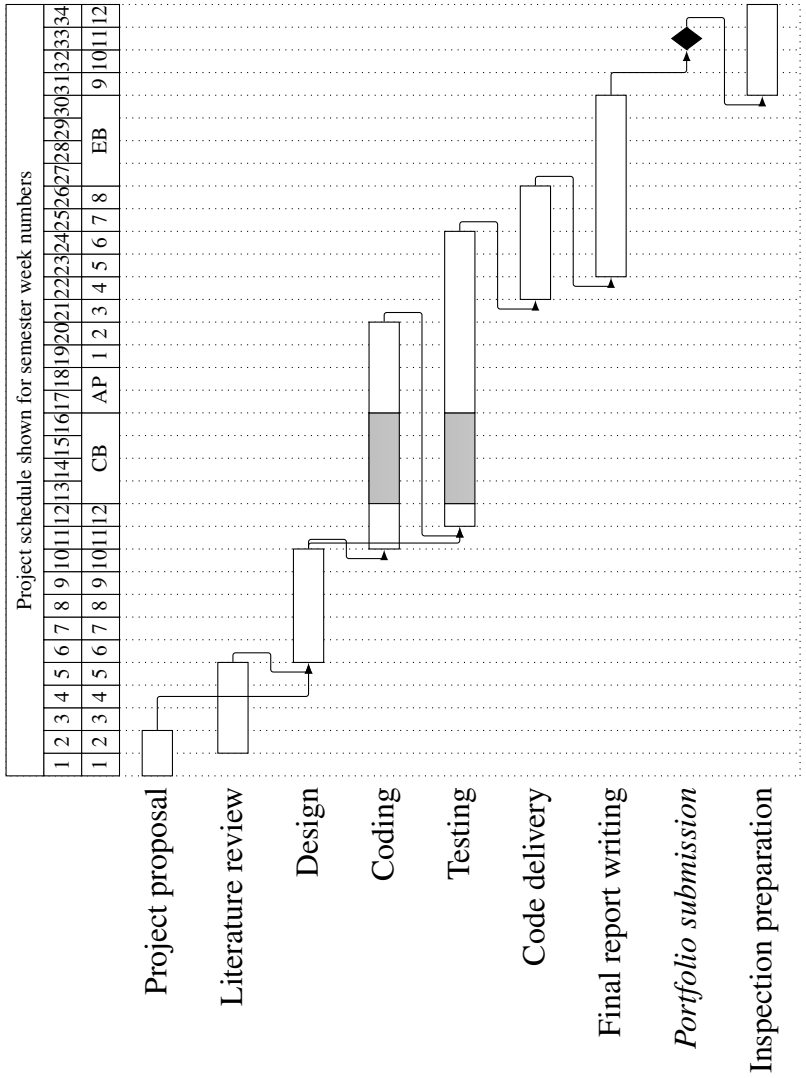

Figure 2: 5.b Modified Gantt Chart

References

Burrows, M. and Wheeler, D. (1994). A block-sorting lossless data compression algorithm. Technical report, DIGITAL SRC RESEARCH REPORT.

Goswami, D., Sultana, N., and Bristi, W. R. (2020). Algorithms for string comparison in dna sequences. In *Proceedings of International Joint Conference on Computational Intelligence*, pages 327–343. Springer.

Hazelhurst, S., Liptak, Z., and Zimmerman, J. (2003). A comparative study of biological distances for est clustering. Technical report, Citeseer.

Lipt´ak, Z. (2018/19). The q-gram distance. *http://profs.scienze.univr.it/~liptak/FundBA/slides/StringDistance2_6up.pdf*.

Melsted, P. and Pritchard, J. K. (2011). Efficient counting of k-mers in dna sequences using a bloom filter. *BMC bioinformatics*, 12(1):1–7.

Roy, R. S., Bhattacharya, D., and Schliep, A. (2014). Turtle: Identifying frequent k-mers with cache-efficient algorithms. *Bioinformatics*, 30(14):1950–1957.

Vinje, H., Liland, K. H., Almøy, T., and Snipen, L. (2015). Comparing k-mer based methods for improved classification of 16s sequences. *BMC bioinformatics*, 16(1):1–13.

Burrows and Wheeler (1994) Goswami et al. (2020) Melsted and Pritchard (2011) Roy et al. (2014) Vinje et al. (2015) Hazelhurst et al. (2003) Lipt´ak (1819)