Bioinformatics Pipeline

Antonio Almeida, UP2015058365

June 18^{th}, 2019

1 Introduction

This report refers to the the third project of the Algorithms for Bioinformatics course. The main goal of the project was to implement a number of procedures common in Bioinformatics work, and aggregate them through a pipeline like structure. The required procedures were: a BLAST[1] implementation, operations for Multiple Sequence Alignment, a UPGMA[2] implementation and respective visualization through phylogenetic trees, and the generation and visualization of a Biological Network.

2 Implementation

2.1 BLAST

The BLAST implementation was based on the code developed during class and the algorithm used follows the main steps described in the course's slides. Word matching is made without the use of a threshold, i.e., only perfect hits are considered. The suggestion mentioned in the project's specification, i.e., to use `best_alignment` N times when looking for N sequences, was not followed. Instead, a more optimized version was implemented, which runs the BLAST algorithm once, and sorts the sequences by score, retriving the top N. Additionally, verifications are made made as to not including sequences of the same species - query sequence included - on the resulting sequences. For the specific database provided, using different W values did not change the results' relative positions, mostly because of the use of perfect matching.

2.2 Multiple Sequence Alignment

The MSA implementation is comprised of two classes, i.e., `MSA`, which handles the general steps of the multiple sequence alignment algorithm, and `AlignmentWrapper`, which represents an instance of an alignment of N sequences and provides useful functionalities, such as computing consensus and calculating alignment scores. The approach taken was to follow the algorithm specified in the course's slides, while taking advantage of the already developed features, namely pairwise sequence alignment functions.

Substitution matrices can be used, the default one being `blosum62`, but others can be generated/provided. Visualization of the alignment was also implemented, alongside the sequences description or species. However, given most sequences used in this context are large, visualization through text is not satisfatory.

2.3 UPGMA

The UPGMA implementation followed the main algorithm specified in the course's slides. However, the implementation took a different path. A single class, i.e., `UPGMA` is used for the component, and it wasn't

[1] https://en.wikipedia.org/wiki/BLAST
[2] https://en.wikipedia.org/wiki/UPGMA

deemed necessary to use a specific class to handle matrix operations (regular lists are used). Trees/clusters are stored in a dictionary, whose keys are labels representing said clusters, e.g., `'(1,(4,5))'`, which are updated on each iteration of the algorithm. Attempts to implement UPGMA using different data structures were made, namely by replacing the distance matrix with lists of tuples, but the results were not satisfatory. The binary tree implementation, i.e., `Tree`, is based on the code developed during class. An improved version of the text visualization method was implemented, alongside one for visualization as a phylogenetic tree.

2.4 Network

The network visualization implementation, i.e., `Graph` was based on the code developed during class. The key for implementing the required functionality was to adapt the classe's constructor to take a matrix as input, i.e., a starting distance matrix from an UPGMA execution. Additionally, Graphviz[3] was used to generate visualization of the graph.

2.5 Pipeline

The main goal with the `Pipeline` class was to provide an easy interface to all the components specified, whether individually or sequentially. With this in mind, a `Pipeline` instance holds the results from an operation, and uses them by default when the next "logical" step is executed by the user. A usage example of the provided API can be seen on `run_me.py`.

The files `blast.txt` (W=3), `msa.txt` (sm=blosum62, g=-3), `upgma.txt`, and the directory `graph` (multiple cut values) contain the results of the various stages of the pipeline, when ran sequentially on the given database.

3 Results

All the required features were implemented. Notable/extra features:

- Used BLAST algorithm is more optimized than the suggested approach;
- MSAs can use different types of substitution matrices, and can be visualized;
- Visualization of UPGMA results as a phylogenetic tree;
- Visualization of networks/graphs through generated images;

4 Conclusions

The main goal of the project was successfully accomplished as the author has learned substantially on the various topics covered by the work, as well as on what may constitute a type of job on the field of Bioinformatics. However, if given more time, further research would've gone into testing different combination of parameters on the various stages.

5 References

No papers were referenced. However, footnotes have been added with hyperlinks for crucial keywords.

[3] https://pypi.org/project/graphviz/