

Giorlando Ramirez

Bio 131 Independent Project

Ritz

5/11/2017

Navigating Phylogenetic Trees using Graphing Algorithms

The motivation behind this project was to create a program that could navigate between species in phylogenetic trees. Phylogenetic trees can be huge, so calculating the evolutionary distance between species that are far away from each other can be difficult. To achieve this, I used Breadth-first search as a means of finding the shortest path in between two species in the tree. The way Breadth-first search works is: the algorithm starts at a beginning node, visits neighbor nodes first before moving on to next level neighbors. It essentially sets up a queue that runs through all possible nodes until it finds its goal node. This is the opposite approach to Depth-first search which goes the full depth of every possible path.

Breadth-first Search, how it works:

Input: A Graph, Starting node in Graph and goal node in Graph.

Output: Goal node. Shortest path back to starting node.

Data

The data I used to test/create the practice data sets I used was retrieved from this website: <http://evolution.genetics.washington.edu/phylip/newicktree.html>. The data is in the “Newick” format, example:

(Bovine:0.69395,(Gibbon:0.36079,(Orang:0.33636,(Gorilla:0.17147,(Chimp:0.19268,Human:0.11927):0.08386):0.06124):0.15057):0.54939,Mouse:1.21460):0.10;.

For the purposes of this experiment I had to convert it to a graph/dictionary pair.

As such: Dataset1 = [['Chimp','A'], ['Human','A']] etc.

Dataset1Dictionary = {'(Chimp','A'):0.19268, ('Human','A'):0.11927} etc.

The purpose of changing the data to this format was so we could use the graph version to find the paths. Then from the paths, we can find the edges and compare them to their dictionary values. This in turn allows us to find the “Evolutionary distance” in between two species in the phylogenetic tree.

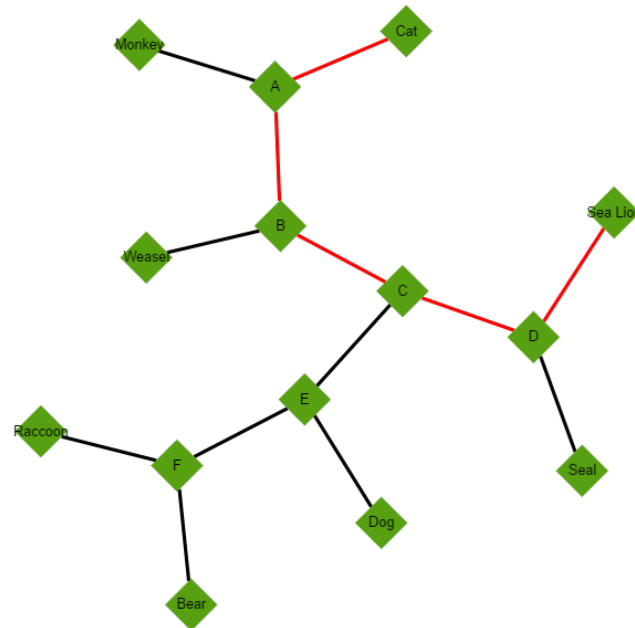
Results

The program works on the practice data sets I used. After given the input of data set, beginning species and goal species it has as an output: the path taken, distance and number of edges. It also gives the user the option to plot the graph to help visualize the path between the two species. Below is an example of the program in motion:

```
Giorlando@G10PC /C/Users/Giorlando/Desktop/Bio131Final
$ py TreeNavigator.py
Hi! :D. Here are the available data sets: Data Set 1, Data Set 2
Choose a dataset(Case and Space Sensitive): Data Set 2
Here is the species list in this data set! :D
List: ['Monkey', 'A', 'Cat', 'B', 'Weasel', 'C', 'D', 'Seal', 'Sea Lion', 'E', 'Dog', 'F', 'Bear', 'Raccoon']
Enter your beginning and goal species to find the distance, number of edges and path between them.
Enter beginning species: Cat
Begin: Cat
Enter goal species: Sea Lion
Goal: Sea Lion
Path Taken: ['Cat', 'A', 'B', 'C', 'D', 'Sea Lion']
Number of Edges: 5
Length(Evolutionary Distance Between Them): 89.35403
Would you like to plot the path in the Graph?(Yes/No):
```

Upon inputting “Yes” into the plotting prompt, the following graph appears on GraphSpace.

Here we can see the whole graph, with the path highlighted in red.



Overall, the results were very positive. The program runs and does exactly what it is supposed to on the practice data. It returns accurate values and uploads a helpful graph to Graphspace.

Conclusions

The program works but has a major limitation. It can take in data, return what is wanted and create nice visuals to help understand what is being done. However, the current state of the data is a major limitation. It can only use the available data sets and adding a new data set would require lots of work by hand. It would certainly be very beneficial and interesting to build off this program by creating a way to improve or automate the use of data. However, it would require creating a program that directly imports large data sets and converts the data into a usable graph/dictionary format. This improvement would make the program much more useful and take away a major limitation. Overall, the project was mostly successful and effective at reaching its goal.