Ben Loisch

CS 251 Project 4

4/9/17

For my synsets, I store the first noun of every synset in an array that holds all the nouns. I do this because when I call sap.ancestor(int, int) later and get the ID that corresponds to a noun in a synset, I can look up the first noun of that synset in my array instantly.

For my synsets, I also store every noun of every synset in a map where the key is a synset noun and the value of that key is an Array List of integers. The Array List is for nouns that have more than one ID. When I'm given a noun to compare to another noun, I use this map to lookup their corresponding IDs.

For my hypernyms, I store the vertices in a directed graph. I do this so I can use my Shortest Ancestral Path class on this data structure. My SAP can perform a breadth first search on the data in this digraph.

My SAP algorithm to compute the shortest ancestral path takes in two vertices v and w. It goes through all vertices in my directed graph and performs breadth first search with v as the source vertex and w as the source vertex. We find the length of the path from v to the current vertex, and w to the current vertex to compute the total length.

The worst-case running time of my SAP.ancestor(int,int) is as follows:

When creating the BreadthFirstDirectedPath object for v or w, the constructor takes O(V + E) time.

We create an object for both v and w, so that would be O(2 * (V + E)).

Next, we go through all vertices and perform bfs() with the two BreadthFirstDirectedPath objects we created for v and w. This will take O(V) time, since the functions inside the for loop that check distance and paths are completed in O(1) and the for loop executes V-times.

Total worst case running time would be O(V + (2 * (V + E)).

The worst-case running time of my SAP.length(int, int) is as follow:

I do exactly the same thing as I did in SAP.ancestor(int, int) except I don't keep track an ancestor.

Total worst case running time would be O(V + (2 * (V + E)).

Total worst case running time of my SAP algorithm (since we output length and ancestor) is:

O(2 * (V + (2 * (V + E))))

The best case running time is exactly same. This is because the implementation is exactly the same. We create two objects, and loop through all vertices. We do this twice, once for SAP.length(int, int) and once for SAP.ancestor(int, int)