Prishi Amin
20998638

ECE 250 Project 3 Design Document

Class Design

1) illegal_exception Class: This handles invalid inputs, specifically when there are capital letters in the classification input strings. This issue of invalid inputs can arise in the methods: insert_classification(), classify_input(), and erase_classification().

2) Node Class: This class is used to represent each node in trie. For each node, this class holds: the classification, a status variable to indicate whether the node is a terminal node or not (terminal nodes are leaves or the end of the classification in this scenario), and a vector that holds the node's children.

3) Trie Class: This class has a root node that is the starting point of the trie. All of the classification nodes stem from this root node. This class contains the methods to handle the possible commands that could be inputted into the program, along with helper functions that support the functionality of the methods.

Function Design

1) Node Class:

- This class has methods to work with the nodes individually. The methods are:
    - **is_terminal_node():** determines if a node is a terminal node or not
    - **set_terminal_node(bool):** Used to update the terminal status of a node based on a boolean input value
    - **node_value():** returns the classification value of the node
    - **set_node_value(string):** assigns a new classification value to the node

2) Trie Class:

- **load_input(string filename):** This initializes the trie based on a .txt file input. The method reads classifications from the file and then splits them into parts. These parts are then inserted into the trie's nodes.
- **insert_classification(string classification):** This method adds new classifications into the trie by splitting the input into parts and then traverses the trie to find out where they fit in the hierarchical structure. It keeps track of duplicates and any invalid entries.
- **classify_input(string input):** Traverses the trie based on the input, collects child classifications, and uses the labelText function for predictions. It prints the classification path along with the predicted class.

- **erase_classification(string classification):** This uses the labelText function from the socket files to traverse the trie and remove specified classifications. While traversing it keeps the terminal status of each node updated and backtracks to remove any nodes that are not needed after deletion of other nodes (i.e. children of parents who have been erased).
- **print_classifications():** This formats all of the nodes in the trie in order to print all of the possible classification paths. This is done recursively using a helper function, so that it meets the time constraint of O(N) time. Each path is formatted with commas and underscores for printing.
- **clear_all_nodes():** This clears the trie of all its nodes and frees all memory. The nodes are deleted recursively using a helper function to meet the time constraint of O(N) time. All nodes are deleted in this process including te root node, but then the root is initialized again once all of the memory is freed from the trie.

Runtime Analysis

Command	**Runtime**	**Analysis**
INSERT	O(n), n is the number of classes in the classification	The classification is split up into parts, then the trie is traversed for each of these parts. Insertion occurs wherever needed. This takes O(n) time since each part of the classification is visited once.
CLASSIFY	O(N), N is the number of classes in the trie	The trie is traversed based on the path inputted. The classifications are taken from the current node's children. The method traverses through each class in the trie to do so, making it O(N) time.
ERASE	O(n), n is the number of classes in the classification	The trie is traversed based on the classification path that is inputted. When the classification is found, the function will backtrack to remove nodes that are no longer needed. The function traverses through the trie for the number of classes in the classification, making it O(n) time.
PRINT	O(N), N is the number of classes in the trie	The trie is traversed recursively in order to get all of the possible paths using a helper function. The method only visits each node once, which takes O(N) time, while doing formatting for the path output takes O(N) time as well.
EMPTY	O(1)	The method checks if the root has no children. Checking the size of the children vector takes constant time.

CLEAR	O(N), N is the number of classes in the trie	All of the nodes in the trie are deleted recursively. The traversal to delete the nodes starts at the root and then every node is visited once to delete it, thus taking O(N) time.
SIZE	O(1)	The number of classifications in the trie is stored and the value is directly returned for this command, which takes constant time since a stored value is being accessed.

UML Diagram