CS/IT 200

Lab 5: 20 Questions

Recommended Due Date: Thursday, October 15, 2020, 11:59pm

Submission

- Submit all code and other materials in a single zip file to the Lab 5 assignment folder on Kodiak.

Goal: Build a system that learns to play a “20 Questions”-like game.

Introduction

Prior to doing any design, make sure to thoroughly review each of the modules which were distributed with this lab.

If anything is unclear about these classes, ask! Your textbook describes these classes in further detail in Sections 8.1-8.3.

Problem

You may have seen online games that purport to be “psychic,” with the ability to correctly guess any item or character that you may be thinking of by asking yes/no questions of the player. Your goal is to write a program that can play this game, in part by learning about a universe of your choice as it plays by asking yes/no questions.

For example, your program might learn about animals by having the following dialogue with its user. (For readability, user responses are shown here in red. The responses also use capital letters, which is not a requirement for this assignment.)

```
Think of an animal, and I will guess it.
Does it have legs? YES
Is it a cat? YES
I win! Continue? YES

Think of an animal, and I will guess it.
Does it have legs? NO
Is it a snake? YES
I win! Continue? YES

Think of an animal, and I will guess it.
Does it have legs? NO
Is it a snake? NO
I give up. What is it? EARTHWORM
Please type a question whose answer is yes for earthworm and no for
snake:
DOES IT LIVE UNDERGROUND?
Continue? YES
```

```
Think of an animal, and I will guess it.
Does it have legs? NO
Does it live underground? NO
Is it a snake? NO
I give up. What is it? FISH
Please type a question whose answer is yes for fish and no for snake:
DOES IT LIVE IN WATER?
Continue? NO

Good-bye.
```

The program begins with minimal knowledge about animals: It knows that cats have legs and snakes do not. When the program incorrectly guesses "snake" the next time, it asks for the answer and also asks for a way to distinguish between snakes and earthworms.

The program builds a binary tree of questions and animals. A YES response to a question is stored in the question's left child; a NO response to a question is stored in the question's right child.

Task

Extend (make a child class of) the `LinkedBinaryTree` class to create a data structure that supports this game. Make use of the protected methods in `LinkedBinaryTree` to modify the tree structure. You may name your class anything you wish, though your module must be named **`twenty.py`**. Additionally, you must have a **`play_game()`** function in the module (not your class) that allows the user to play when it is called.

When the game ends, you should offer the player the opportunity to save the data in the tree so that they do not have to re-train your program from scratch every time. When starting the game, you should give an option to load a saved game. Make use of the `save_tree()` and `load_tree()` methods inherited from `BinaryTree` to save and load the game from a file. You should not modify these methods, just call them. Note that the files created here are binary files that are not human-readable. The table below shows examples of using these methods:

Save a tree to gametree.dat	`mytree.save_tree('gametree.dat')`
Load a tree from gametree.dat	`mytree = BinaryTree.load_tree('gametree.dat')`

What to Submit:

- Your code (`twenty.py` and any other necessary modules, including modules given as class materials)
- The input/output of you training the system from scratch (like the example above), copied from the console.
- A saved version of the tree that was created from the submitted training input/output.

Rubric

Grade	Overall	Code/Structure	Execution	Output
4	Sums up to 10, with at least a 3 in Code/Structure and Execution	Classes, inheritance, and functions set up as described. Proper style and structure. Code is well commented.	Algorithm works perfectly	Perfect match between saved tree and submitted training input/output
3	Sums up to 8 with at least a 3 in any two categories, and at least a 2 in Code/Structure	Issues with one of the following: class structure, functions, coding style, comments	Minor error in algorithm or exceptions raised in rare corner cases	Saved tree and input/output don't quite match
2	At least a 2 in Code/Structure and another category, and at least a 1 in all categories	Few/no comments -AND- Issues with one of the following: class structure, functions, coding style	Exceptions or unexpected behavior in multiple cases -OR- Only allows you to play one round before resetting the tree	Serious disagreement between training and save file -OR- Only one of training and save file submitted (or both submitted, but unrelated)
1	At least a 1 in Code/Structure and at least a 1 in Execution	Few/no comments -AND- Issues with two of the following: class structure, functions, coding style	Exceptions or unexpected behavior in multiple cases -AND- Only allows you to play one round before resetting the tree	Training output and save file are limited (e.g. only 1 round, or baseline tree)
0		Incomplete or unrelated work, syntax errors	Syntax errors, or exceptions thrown frequently or unavoidably	No output or save file given