11-28-2016

Computational Model of Intelligent Behavior

Lindsey Brianna Harris

Computational Modeling of Intelligent Behavior

by

Lindsey Brianna Harris

An Honors Capstone
submitted in partial fulfillment of the requirements
for the Honors Diploma
to

The Honors College

of

The University of Alabama in Huntsville

Nov 28, 2016

Honors Capstone Director: Dr. Daniel Rochowiak
Associate Professor, Computer Science
Director, Collaborative Learning Center

Lindsey B. Harris	12/02/2016
Student	**Date**
[illegible]	11/29/2016
Director	**Date**
[illegible]	11/29/2016
Department Chair	**Date**
[illegible]	12/9/16
Honors College Dean	**Date**

HONORS COLLEGE
THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Honors College
Frank Franz Hall
+1 (256) 824-6450 (voice)
+1 (256) 824-7339 (fax)
honors@uah.edu

Honors Thesis Copyright Permission

This form must be signed by the student and submitted as a bound part of the thesis.
In presenting this thesis in partial fulfillment of the requirements for Honors Diploma or Certificate from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department, Director of the Program, or the Dean of the Honors College. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this thesis.

Lindsey B. Harris
Student Name (printed)

Lindsey B. Harris
Student Signature

12-02-2016
Date

Table of Contents

Dedication:

This Capstone Project is dedicated to my parents, namely my mother, Angela Harris, who inspired me to become a Computer Science major and to explore programming and other such computer science skills.

I dedicated this also to the love of my life, Matthew Pittenger, who inspires me to work hard and solve problems on a regular basis. Without whom, my problem-solving skills would not be nearly as cultivated. Thank you for solving so many challenges alongside me.

Lastly, I dedicate this to my fellow Computer Science majors, who hopefully will find this matter to be useful and/or inspirational as they continue in their studies.

Abstract

The problem of modeling intelligent behavior deals with varied probabilities of actions as well as patterns of action distribution in behavioral trees. It is simple to model a perfectly rational entity, which makes the rationally correct choice all of the time. However, because humans do not live perfectly rational lives, a perfectly rational model is not very realistic. Therefore, modeling human-like behavior accurately must also take into account the irrational things that humans do that still seem to make sense. Some of the “irrationalities” are based on social norms, while others are based on habits. Utilizing a number of iterative steps, the branch of action that the model takes could change based on monitoring that occurs during the previous steps that the model has taken. With that in mind, the purpose of this project is to research the probability trees as behavioral structures in order to produce a reusable software component that realistically models human-like artificial intelligence.

Introduction

Artificial intelligence (AI) and behavioral modeling have become major topics of interest in the past few years, due to the vast range of applications it has. Typically, artificial intelligence is associated with video games, however, there are several other applications as well. According to Treanor, “AI is used to achieve ends: automated classification of data, object recognition, planning, and language generation are just a few AI capacities” (4). Here, the objective of AI is to model human behavior in decision-making, i.e. given a choice, the model should select an option from a set of options provided by the choice. When viewing AI as a behavioral model for humans, it is important to consider factors such as habit, aversion, and irrational spontaneity when making choices.

Humans are creatures of habit, a habit being an act that is repeated frequently over the course of time. The more often a person performs an action, the more likely that person will perform that same action in the future. The exception is when negative consequences create an aversion to the action, aversion being “a strong feeling of dislike … [which causes] a turning away or preventing” (“aversion”). With that in mind, it would seem simple to create an AI that models human behavior by selecting beneficial options and avoiding harmful options. However, humans can be irrational and spontaneous, which is what helps enable them to have personality. The question that then arises is how can one model habits, aversions, and spontaneity without giving the appearance of completely random selection? The proposed solution is to use probability trees in the AI choice selection process.

Project Description

The purpose of this project is to explore the usability of a probability tree structure for behavioral AI and to demonstrate the structure's effectiveness in an artificial intelligence setting. This will be accomplished through a series of iterative program designs that will build on each other and will eventually result in a finalized proof of concept program that will demonstrate the probability tree structure's usability. As a proof of concept, a random story generator program, which uses the probability tree structure to select options from a given story line, will be designed and implemented. The iterations of this project will include a preliminary overview and exploration of the structures to be used (iteration 0), an implementation of the "simple case", which will be a time-independent implementation that will select options from the same choice repeatedly in order to test the basic structure (iteration 1), and an implementation which will include a one-step memory (iteration 2). Finally, there will be a demo story tree that can be used by the generator to demonstrate the iterations of the project. This project will be programmed in Java, however, it must be noted that the implementation will be covered within the appendixes. The goal of this document is not to discuss the implementation of the project, but rather to share the concepts which others can build upon.

Preliminary Structures

To understand the implementation of this project, it is crucial to first understand the structures that are being discussed throughout the text, namely the object of our study: the probability tree.

The Basic Tree

Anyone who has much experience with programming data structures should be aware of the tree data structure. Trees are hierarchical structures consisting of branches and nodes, where a node is a structure that holds data and a branch is a connection between two nodes. The “root” node is the base of the tree, and all of the other “leaf” nodes derive from it. The nodes directly connected to the root node are called the root’s children, and each of these “child” nodes can have child nodes of their own. A node with children connected to it is called a “parent” node. For a visual example of what a tree looks like, see Figure 1 below.

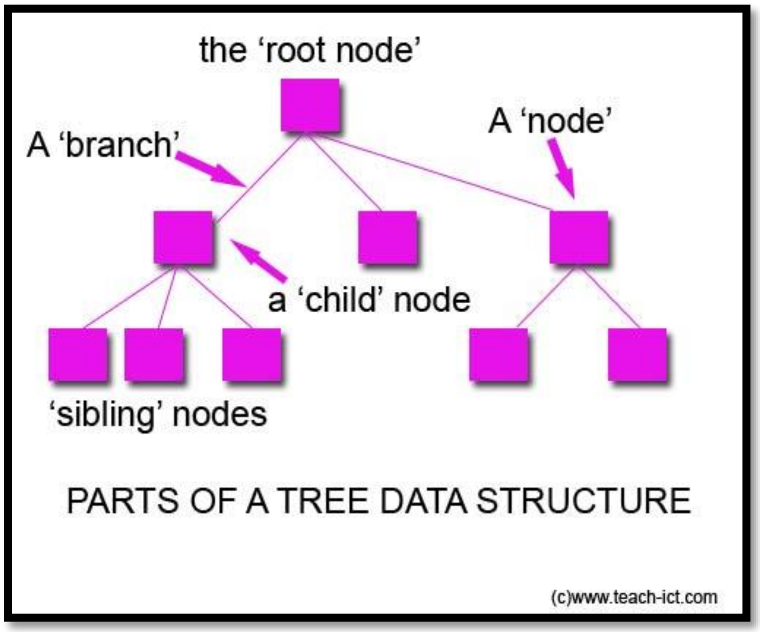


Figure 1 (Google Images)

Trees are used in a variety of applications, and are often used to store data that can be sorted in some hierarchical format. Often times, this makes it much easier for programmers to search for specific pieces of data more quickly, and it also allows users to make varying paths from the root to an ending leaf node. The path application is what will be used here to create the different stories in the story generator program. Each parent node will be a “choice” that the program can make, while the child nodes will each be an option that results in another choice. While this is quite useful, it is not enough to simply have a basic tree of choices to create a believable AI. To add some variability that will make the AI more believable, a special type of tree called a probability tree will be used for the story generator program.

The Probability Tree

A probability tree is a tree with multiple children whose nodes contain probability data. This probability data can be utilized by the probability tree’s extra functionality to determine what percentage of the time a given child node ought to be selected. For example, let us look at a simple tree where parent node P has branches to its child nodes A, B, and C. Let us also say that the user wants to randomly select one of the children. Using a normal tree, random selection would have to be implemented outside of the tree structure, and would likely give equal opportunity to each of the child nodes because there is nothing in the tree that conveys any sort of priority. In a probability tree, however, the functionality allows the user to select nodes based on the priority (probability) of the given node.

For example, looking at a tree with parent node P, let us say that node P is the choice of what to do when you are inside of a burning building and you see someone trapped behind a beam that you are capable of moving. Let us say that choice A is to help the person get out from behind the beam and then leave, choice B is to leave the building and to tell a firefighter that you saw a person trapped and where you saw them, and choice C is to leave the building and say nothing because the firefighters have it handled. If attempting to design an AI that selects one of these options using a normal tree structure, there is no way to determine which path should be chosen if one should be chosen more often than another. The only way to implement the selection process with a normal tree is to have a selector implemented outside of the tree. This selector then has two options: it can either select randomly, giving equal opportunity to each of the child nodes, which is highly unpredictable, OR it can select unevenly, giving weight to each of the options. However, the weighted approach would have to be based on the order of the nodes, and could not be based in the value of the node itself. This is because the weights are not implemented within the nodes themselves, but within the selection process. In other words, a weighted implementation might select node A 50% of the time, node B 30% of the time, and node C 20% of the time, but it would give those same probabilities to every choice that was made in that order, because the weights are tied to the selection process instead of to the individual option nodes.

With a probability tree, we can say that node A (choice A) has the probability value of 0.7, node B (choice B) has the probability value of 0.2, and node C (choice C) has the probability value of 0.1. When the user goes to select a child of P, now there is data that can be used such that you rescue the person 70% of the time, leave and tell a firefighter about the person 20% of the time, and ignore the person only 10% of the time. Given another choice, you would be able to set node A’s probability value to 15%, node B’s probability value to 40%, and node C’s value to 45%. As you can see, this enables the tree to determine the value of each choice, and simplifies implementing a useful and adaptable artificial intelligence. Other methods, such as completely random selection or weighted selection carry with them simplicity as well, but they are not nearly as flexible or realistic as the probability tree implementation. Case in point, with the firefighter example, a normal tree would make all characters utilizing the tree approximately the same, whereas with the probability tree, you could have some characters more likely to be a hero, or some to be more of an oblivious person. Therefore, a probability tree can be used to add personality to different AI characters, which helps make the characters realistic. This added realism makes the probability tree structure ideal over some of the other AI implementations. For probability tree implementation details, see Appendix A.

The Simple Case

One of the simplest examples of using a probability tree is the case where a single choice is made repeatedly. Selecting one option within a choice has no effect on selecting an option the next time that same choice is made. Mathematically, this can be equated to an example commonly used in Probability and Statistics courses, the urn problem.

The Urn Problem

An urn problem is a probability problem that uses an urn (hence the name) and a selection of different colored marbles. The marbles are all put into the urn and one marble is randomly selected out of it, as can be seen in Figure 2 below.

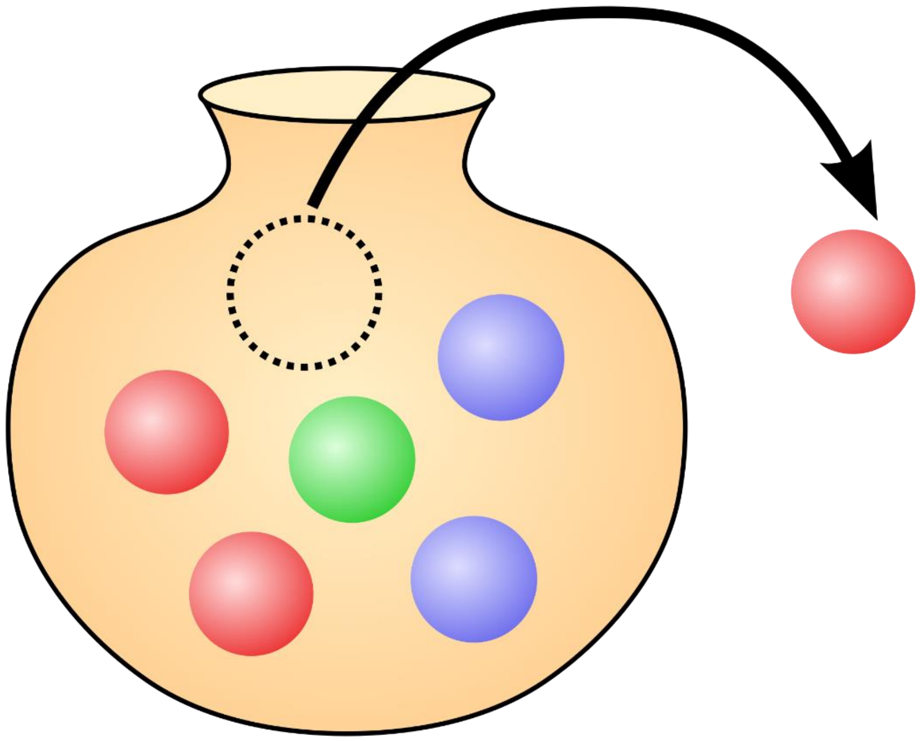


Figure 2 (Google Images)

If the marble is put back into the urn before another marble is selected, it is said that this is an urn problem with replacement. With replacement, it is equally likely to pull a marble of the same color out of the urn as before. Our simple case works much like the urn problem with replacement. When we have a choice with options to select, every option has the same likelihood of being chosen each time the choice is made. In this case, the choice is the urn and each option is a different color of marble. The probability of each option correlates to the number of each type of marble in the urn.

Simple Case in Action

Let's apply this simple case to a sample probability tree to see what this type of selection process looks like. Take, for instance, a choice that involves traveling down a path. When you come to a split in the path, this case consists of 3 options: turn left, turn right, or go straight. The left path is broad and flat, the right path leads into a questionable part of a dark forest, and the center path is incredibly steep, but all pathways lead to your destination. The left pathway is the farthest distance, the steep pathway is the middle distance, and the dark woods are the shortest distance. Let us also say that you have difficulty with steep hills and are slightly afraid of the dark. On the average day, you are 50% likely to take the path on the left, 30% likely to go straight up the steep path, and 20% likely to venture into the dark wood.

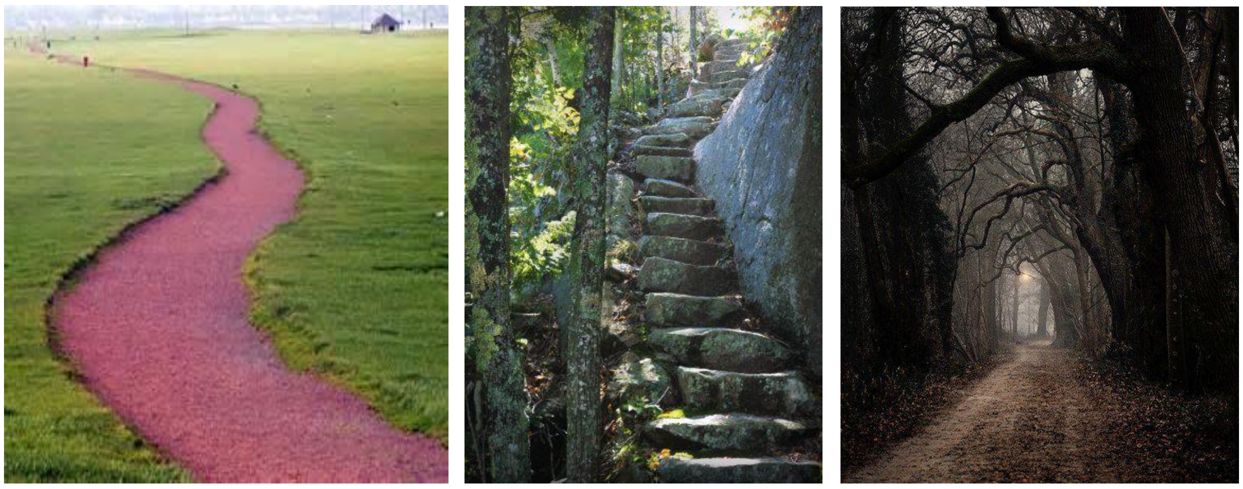


Figure 3 (Google Images)

If you come to this route on 10,000 different days, the chances of taking each path does not change from day to day. So over the course of the 10,000 days, you should take the left path ~5,000 times, the straight path ~3,000 times, and the right path ~2,000 times. This is great for simple choice-making, and makes characters slightly more believable than complete random selection when applied to artificial intelligence. However, this method would make all characters using the same probability look approximately the same. To give characters a little more personality, it is necessary to allow them to build habits, which will then differentiate them from other characters using the same probability tree. To do this, we need to examine the second iteration of the project, the one-step memory.

One-Step Memory

Conceptually, the one-step memory problem is exactly what it sounds like. With this case, when making a choice, selecting an option will affect the probabilities of the options the next time the choice is made. The goal of the one-step memory is to simulate making decisions based on previous decisions. This could be applied to forming a habit or avoiding unpleasant scenarios. Some scholars view a choice as "being about *choosing* or *rejecting* options, in order to 'manipulate attention to positive and negative features…We tend to choose options because of their positive features but reject them because of their negative features'" (Lockton 7).

Forming a Habit

Human beings are often said to be creatures of habit, where habit is defined as "an acquired behavior pattern regularly followed until it has become almost involuntary" ("habit"). For example, washing your hands after going to the bathroom is a habit. Oftentimes, people see an individual's habits or mannerisms as a part of that individual's personality. With washing one's hands, people see it as a good habit and may look down upon any person that does not wash their hands when they leave the restroom. The more often you wash your hands, the more likely you are to continue washing your hands. However, if a person does not wash their hands in the restroom regularly, they are less likely to do so regularly in the future. According to Lockton, "Many behaviors relevant to environmental and social impact are the result of habits formed over time" (14). Therefore, when trying to construct a believable AI, it is important to be able to incorporate habits into the decision-making algorithm. In order to accomplish this, when one option of a choice is selected one time, it should be more likely for that option to be selected again the next time that choice is made.

Avoiding Unpleasant Scenarios

Sometimes there are choices that end badly. Humans do not repeat every single choice that they make—especially if selecting an option results in an unpleasant outcome. Lockton says that “people prefer to take risks to avoid a loss” (4). In these cases, a habit could be formed, but oftentimes people are less likely to select an option that gave them a previously bad experience. For example, if a person gets food poisoning at a restaurant, the likelihood of them returning to that restaurant the next time they go out to eat is significantly smaller than if they had not received food poisoning. Similar to the concept of forming habits, it may be desirable to apply this concept to our AI to make it more believable. To do this, when an undesirable option is selected, then the next time that choice is made, it should be less likely for that option to be selected again.

Bounding Method

Separately, it is simple to create a habit-building and an avoidance algorithm, but how do we effectively combine these concepts to work together when one is adding to the probability of the event and one is subtracting from it? One method of altering probabilities is bounding. Using upper and lower bounds can enable a shift in the dynamic of the selection process. The lower bound can set the lower limit for the selected option the next time a choice is made, and the upper bound can set off the block for the new probabilities. For example, say you have a choice with three options, A, B, and C, as laid out in Figure 3 on the next page.

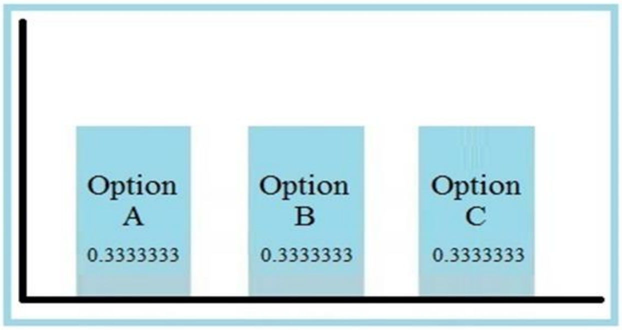


Figure 3

Each of these options has an equal probability of being selected. If we want to change the probabilities to build a habit or an aversion, then we introduce upper and lower bounds. Let Option A be the option that is selected. If the end goal is to build a habit, then the probability for Option A should increase. To do this, the lower bound is positive, and the upper bound is set greater than or equal to the probability of A plus the lower bound, as can be seen in Figure 4 below.

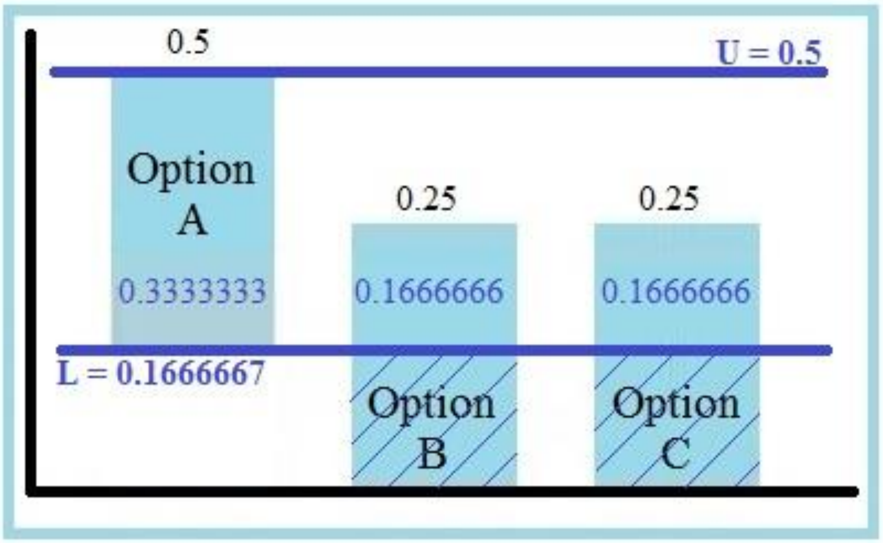


Figure 4

Here, we set the lower bound to 0.1666667, which cuts through the middle of options B and C. Option A builds on top of that, and here we set the upper bound equal to the probability of A plus the lower bound, which is 0.5. This leaves the probabilities of A, B, and C at 0.3333333, 0.1666666, and 0.1666666 respectively. As 0.3333333, 0.1666666, and 0.1666666 do not add up to 1, the probabilities are normalized, which brings the new probabilities to A = 0.5, B = 0.25, and C = 0.25. Because 0.5 is greater than A's previous probability of 0.3333333, this effectively builds a habit when A is chosen.

If instead, we wanted to form an aversion to an option, we could set the lower bound as a negative value. Setting the lower bound as a negative value would subtract from probability of the selected option, and would then act as though the lower bound was zero for the remainder of the options in the choice. An example of this can be seen in Figure 5 below.

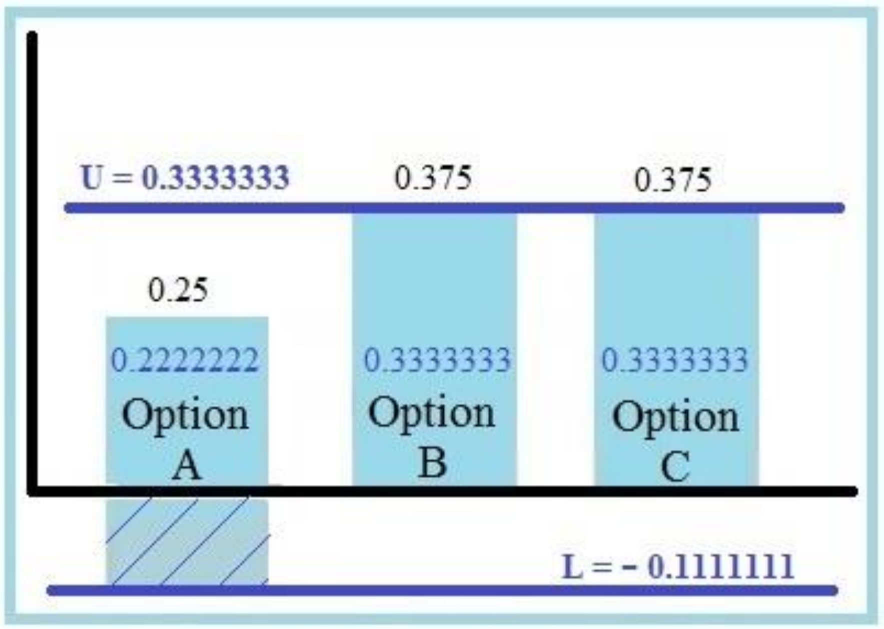


Figure 5

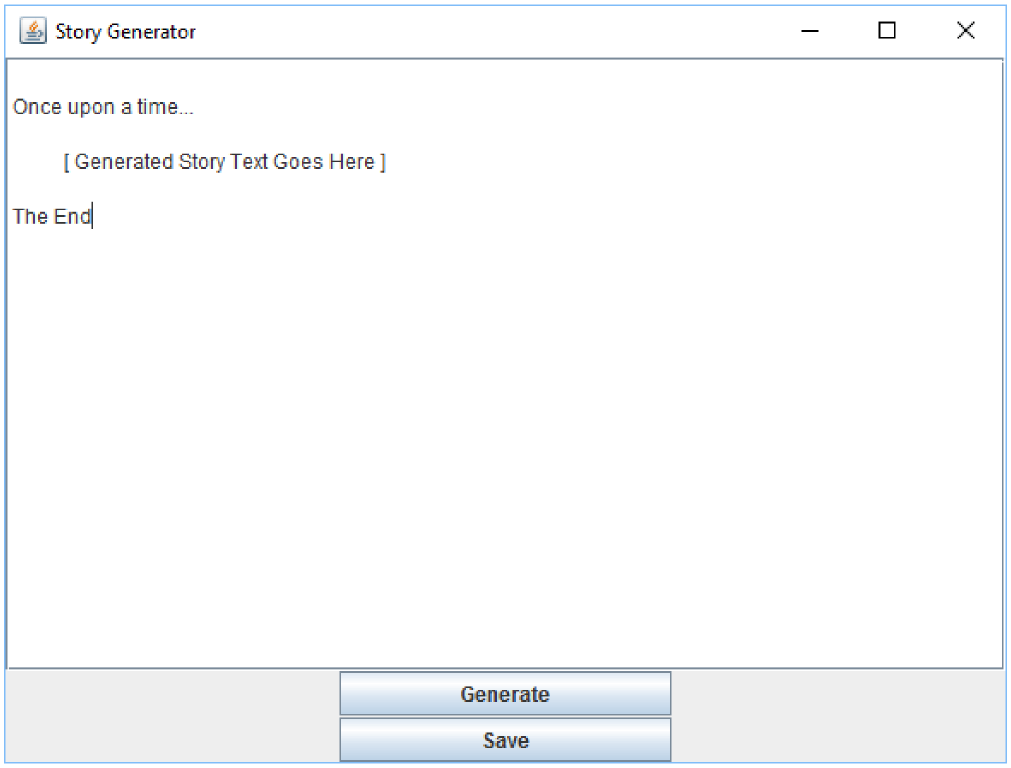
Here, we set the lower bound to -0.1111111, which subtracts 0.1111111 from probability A. By setting the upper bound at 0.3333333, this leaves the probabilities for A, B, and C at 0.2222222, 0.3333333, and 0.3333333 respectively. After normalizing, this results in final probabilities of A = 0.25, B = 0.375, and C = 0.375. This made the probability of A go down, and the probabilities of B and C go up, effectively creating an aversion to Option A. By giving each option a different upper and lower bound, this enables the user to create any number of combinations of habits and aversions that can be used within a character-building AI.

Story Generator Application

Now that we've spent some time using the probability tree structure and how we can apply it to artificial intelligence, let us examine the focal point of the project: the Story Generator Application. The primary goal of this project is to explore the probability tree structure and apply it to AI to create believable characters. The Story Generator Application seeks to fulfill this goal by utilizing the probability tree to generate varying stories based on the likelihood of certain events occurring. Ideally, this application will use the one-step memory to form believable habits for the characters involved.

What it Looks Like

The Story Generator Application from the user's standpoint is fairly simple, consisting of a text field, a generate button, and a save button, which can be seen in Figure 6 below.



Once upon a time...

[Generated Story Text Goes Here]

The End

Figure 6

How it Works

The Story Generator Application uses a preset probability tree, which contains all of the story scene information. When the user clicks the “Generate” button, the application moves through the probability tree scenes and displays the scene text for each of the chosen scenes. This text will reset every time the user clicks the “Generate” button, and a new story will be generated. The generation algorithm will utilize the one step memory within the probability tree selection process to string scenes together for the story. The setup of the probability tree will determine how the one-step memory affects the outcome of the story. For implementation details, see Appendix B.

Future Improvements

The Story Generator application itself is a fairly simple application designed for the purpose of illustrating the probability tree structure with AI. Some suggested improvements of this application include loading a custom Probability Tree file, adding multiple characters, and adding graphics for story illustrations.

Custom Probability Trees

Adding the capability to load a custom probability tree into the Story Generator application would enable users to load a multitude of trees to the story generator, which would have the potential to create any story that they wanted. A custom tree could contain any new scenes with varying probabilities and bounds with which to experiment. This would enable the creation of varying AIs as well as a multitude of characters that the user could create.

Multiple Characters

Adding the ability for multiple characters to utilize a probability tree of choices would further the diversity of stories by enabling multiple characters to make choices that alter the possibilities for the end scenarios. This would create more believable AI worlds which have added depth and variability by an exponential factor every time a new character was added to the tree. Characters also could use varying trees that affect each other. For example, if two characters are talking and one character leaves, then the other character would no longer be able to choose “talk to” to interact with the other character. This feature would be complicated, but would also give a realistic feel to a story.

Story Illustrations

Adding graphics to the Story Generator application would further the story experience. This would give more of a movie feel to the Story Generator application, and would reduce the repetitive nature of repetitive decisions. Instead of reading repetitive paragraphs, it would be easier to watch a character repeat an action visually. For example, if reading “He trained hard.” several times in a row, the sentence loses interest. However, when watching graphics, it is easier to see the big picture of the story instead of scrolling through text. Watching a character train hard repeatedly shows dedication and would be easier for the user to see the outcome without tiring as easily of the repetition.

Demo

To demo the Story Generator Application, we will utilize the following probability tree:

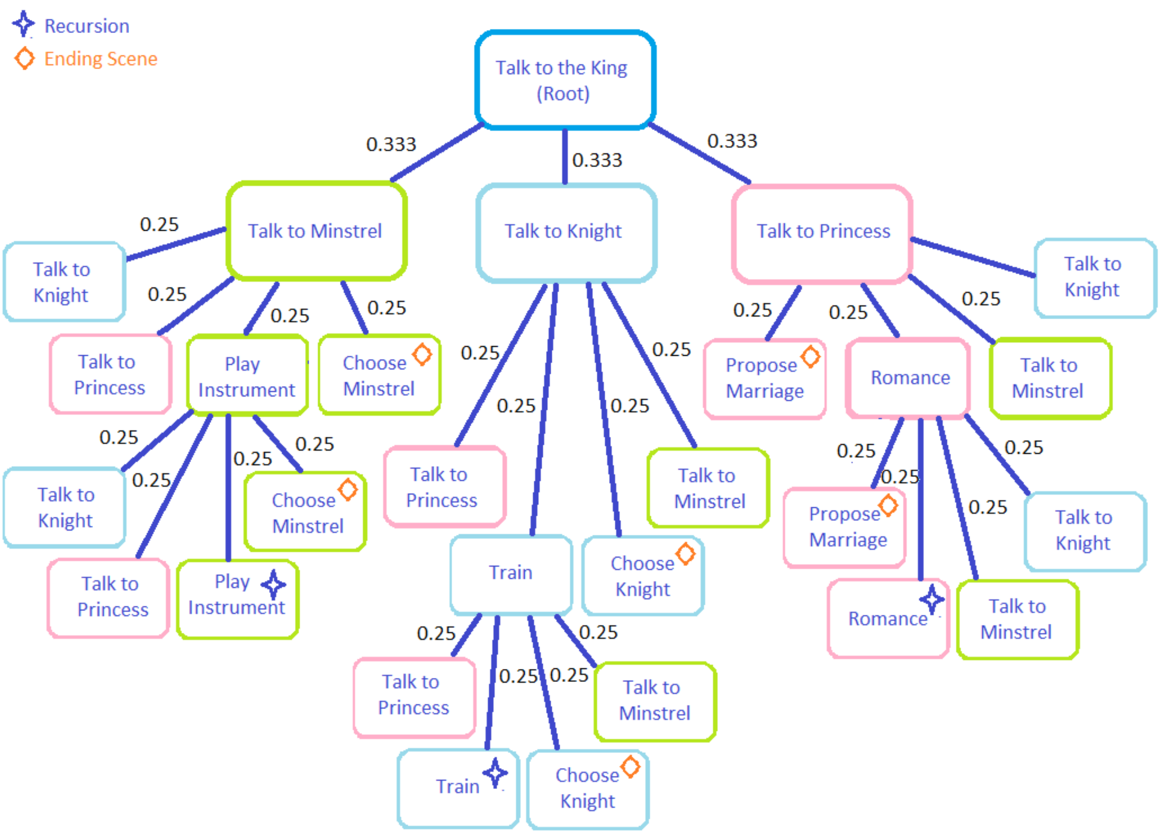


Figure 7

This visualization includes the nodes and probabilities. There are 13 nodes used in this demo tree, and each starts with an equal chance of selection. Some of these nodes are recursive, which will allow the one-step memory to really expand bias towards certain choices. Other nodes are endpoints for the story. The upper and lower bounds applied to each node can be seen in Table 1 on the next page.

Table 1. Demo Tree Bounds

Choice Node	***Initial Probability***	***Lower Bound***	***Upper Bound***
Talk to the King (Root)	1.0	0.0	1.0
Talk to Minstrel (1st level)	0.3333333	0.05	1.0
Talk to Knight (1st level)	0.3333333	0.05	1.0
Talk to Princess (1st level)	0.3333333	0.05	1.0
Talk to Minstrel (2nd level)	0.25	0.05	1.0
Play Instrument	0.25	0.05	1.0
Choose Minstrel	0.25	0.05	1.0
Talk to Knight (2nd level)	0.25	0.05	1.0
Train	0.25	0.05	1.0
Choose Knight	0.25	0.05	1.0
Talk to Princess (2nd level)	0.25	0.05	1.0
Romance	0.25	0.05	1.0
Propose	0.25	0.05	1.0

As can be seen in Table 1, the bounds provide a habit-building structure for the demo tree.

Demo Story Outline

The demo story follows a young man who has grown up in the king's palace. The king has decided that it is time for his ward to decide what he wants to do with his life. Thus, the young man has to talk to the court minstrel and the captain of the guard to determine whether he would prefer to be an entertainer or to defend the kingdom. There is also a princess visiting from another kingdom who fancies his eye. If he marries her, he could take on the duties of a prince. He just has to decide what he wants by the end of the day. Given this scenario, there are a number of interactions that can occur, as outlined in Figure 7. Based on those interactions, eventually one of the given results of becoming a minstrel, becoming a knight, or marrying a princess will occur, and the story will come to an end.

Demo Results

This demo produced a variety of stories, some short and straightforward, others longer. The shorter stories frequently by-passed the one-step memory process by selecting the end scene quickly without training, practicing, or romancing the princess. Longer stories that did select training, practicing, or romance nodes tended to repeat those actions. Table 2 outlines how many times each option was selected per run below.

Table 2. Demo Results

Run	*Talk to the King (Root)**	*Talk to Minstrel*	*Practice Instrument*	*Choose Minstrel**	*Talk to Knight*	*Train*	*Choose Knight**	*Talk to Princess*	*Romance Princess*	*Marry Princess**
1	1	1	-	-	1	2	-	1	-	1
2	1	-	-	-	-	-	-	1	-	1
3	1	1	-	1	-	-	-	-	-	-
4	1	1	-	-	-	-	-	1	-	1
5	1	-	-	-	1	-	-	1	-	1
6	1	1	-	-	-	-	-	2	4	1
7	1	-	-	-	-	-	-	1	2	1
8	1	-	-	-	-	-	-	1	-	1
9	1	-	-	-	-	-	-	1	2	1
10	1	1	2	1	1	1	-	1	1	-
11	1	1	1	-	2	-	1	1	1	-
12	1	-	-	-	-	-	-	1	-	1
13	1	3	3	-	3	-	1	1	2	-
14	1	1	1	1	-	-	-	-	-	-
15	1	1	-	-	-	-	-	1	5	1
16	1	1	-	-	2	1	1	2	-	-

Table 2. Demo Results (Continued)

Run	Talk to the King (Root)*	Talk to Minstrel	Practice Instrument	Choose Minstrel*	Talk to Knight	Train	Choose Knight*	Talk to Princess	Romance Princess	Marry Princess*
17	1	1	-	-	1	1	1	-	-	-
18	1	1	-	-	2	1	1	1	1	-
19	1	2	1	1	2	-	-	1	-	-
20	1	2	5	-	2	-	1	-	-	-
21	1	-	-	-	1	1	1	-	-	-
22	1	1	-	-	2	3	1	-	-	-
23	1	1	-	1	1	-	-	1	-	-
24	1	-	-	-	1	1	1	1	2	-
25	1	-	-	-	-	-	-	1	-	1
26	1	2	1	-	1	-	-	3	1	1
27	1	1	-	1	-	-	-	-	-	-
28	1	1	-	-	1	-	1	-	-	-
29	1	1	-	1	-	-	-	-	-	-
30	1	3	3	1	1	3	-	2	-	-
31	1	-	-	-	-	-	-	1	1	1
32	1	-	-	-	1	-	1	-	-	-
33	1	1	-	-	4	-	-	4	2	1
34	1	3	1	-	3	4	1	-	-	-
35	1	2	-	1	-	-	-	1	-	-
36	1	1	-	1	1	-	-	-	-	-
37	1	2	-	-	1	1	-	2	3	1
38	1	1	-	1	1	1	-	1	-	-
39	1	2	3	-	2	2	-	3	-	1
40	1	-	-	-	1	2	1	-	-	-
41	1	1	1	-	1	-	-	2	-	1
42	1	1	-	1	-	-	-	1	-	-
43	1	-	-	-	1	1	1	1	-	-
44	1	-	-	-	-	-	-	1	-	1
45	1	-	-	-	1	-	1	-	-	-
46	1	1	-	1	-	-	-	1	-	-
47	1	2	5	-	1	-	-	1	2	1
48	1	-	-	-	1	-	-	1	-	1
49	1	-	-	-	1	-	1	-	-	-
50	1	1	-	1	-	-	-	-	-	-
Avg Repeats	0.00	0.39	1.25	0.00	0.45	0.67	0.00	0.35	1.07	0.00

Table 2 illustrates a sample of how many times each option was selected by the story generator, and it also shows how many times on average each choice was repeated after it was selected. As could be expected from the demo tree, the secondary recursive options were repeated more frequently than the primary recursive options. This created believable habits, while also adding some unpredictability to the mix. The order of each of the demo runs is as follows:

1. Minstrel → Knight → Train x2 → Princess → Marry Princess
2. Princess → Marry Princess
3. Minstrel → Choose Minstrel
4. Minstrel → Princess → Marry Princess
5. Knight → Princess → Marry Princess
6. Princess → Romance → Minstrel → Princess → Romance x3 → Marry Princess
7. Princess → Romance x2 → Marry Princess
8. Princess → Marry Princess
9. Princess → Romance x2 → Marry Princess
10. Princess → Romance → Knight → Train → Minstrel → Practice x2 → Choose Minstrel
11. Knight → Princess → Romance → Minstrel → Practice → Knight → Choose Knight
12. Princess → Marry Princess
13. Princess → Romance x2 → Minstrel → Knight → Minstrel → Practice → Knight → Minstrel → Practice x2 → Knight → Choose Knight
14. Minstrel → Practice → Choose Minstrel
15. Minstrel → Princess → Romance x5 → Marry Princess
16. Minstrel → Princess → Knight → Train → Princess → Knight → Choose Knight

17. Minstrel → Knight → Train → Choose Knight
18. Knight → Minstrel → Princess → Romance → Knight → Train x2 → Choose Knight
19. Knight → Princess → Minstrel →Practice → Knight → Minstrel → Choose Minstrel
20. Minstrel → Practice → Knight → Minstrel → Practice x4 → Knight → Choose Knight
21. Knight → Train → Choose Knight
22. Knight → Train x2 → Minstrel → Knight → Train → Choose Knight
23. Princess → Knight → Minstrel → Choose Minstrel
24. Princess → Romance x2 → Knight → Train → Choose Knight
25. Princess → Marry Princess
26. Knight → Princess → Minstrel → Princess → Romance → Minstrel → Practice → Princess → Marry Princess
27. Minstrel → Choose Minstrel
28. Minstrel → Knight → Choose Knight
29. Minstrel → Choose Minstrel
30. Knight → Train x3 → Minstrel → Princess → Minstrel → Practice x2 → Princess → Minstrel → Practice → Choose Minstrel
31. Princess → Romance → Marry Princess
32. Knight → Choose Knight
33. Minstrel → Knight → Princess → Knight → Princess → Romance → Knight → Princess → Romance → Knight → Princess → Marry Princess
34. Minstrel → Knight → Minstrel → Knight → Train → Minstrel → Practice → Knight → Train x3 → Choose Knight
35. Minstrel → Princess → Minstrel → Choose Minstrel
36. Knight → Minstrel → Choose Minstrel

37. Minstrel → Princess → Romance x3 → Minstrel → Knight → Train → Princess → Marry Princess
38. Princess → Knight → Train → Minstrel → Choose Minstrel
39. Princess → Minstrel → Practice x2 → Princess → Knight → Train x2 → Minstrel → Practice → Knight → Princess → Marry Princess
40. Knight → Train x2 → Choose Knight
41. Minstrel → Practice → Princess → Knight → Princess → Marry Princess
42. Princess → Minstrel → Choose Minstrel
43. Princess → Knight → Train → Choose Knight
44. Princess → Marry Princess
45. Knight → Choose Knight
46. Princess → Minstrel → Choose Minstrel
47. Minstrel → Practice x2 → Knight → Minstrel → Practice x3 → Princess → Romance x2 → Marry Princess
48. Knight → Princess → Marry Princess
49. Knight → Choose Knight
50. Minstrel → Choose Minstrel

This shows a fair amount of diversity, while also illustrating how selecting a repeating option increases the likelihood of selecting that option. For samples of the demo story results, see Appendix C.

Conclusion

The probability tree is a good way to model human behavior in artificial intelligence by providing the ability to create habits, aversions, and diverse personality. The largest difficulty in utilizing the probability tree for AI is constructing the tree itself properly with the right numbers to get the desired results, but that should not be terribly difficult for a mathematician. This project illustrated that the use of probability trees is viable for creating a believable AI model, given the proper bounds and probability settings, and the Story Generator application successfully created a diverse range of stories using the probability tree. There are some improvements that could be made to this project, such as the capability of adding custom probability trees, updating scenes such that they can call a function when selected, adding traits to characters, and adding a visual aid or animation to make the stories more interesting. While there are some definite improvements that can be made, the end goal of demonstrating viability for the probability tree structure was achieved, and it will be interesting to see how probability trees are used in artificial intelligence as development of these structures continue.

References

Lockton, Dan. “Cognitive biases, heuristics and decision-making in design for behaviour change.” http://danlockton.co.uk. 2012. (Working Paper)

Mark, Dave. *Behavioral mathematics for game AI*. Boston, MA: Course Technology Cengage Learning, 2009.

Paliath, Vivin. "Generic (n-ary) Tree in Java." Vivin.net. N.p., 30 Jan. 2010. Web. 11 Nov. 2016. http://dspace.brunel.ac.uk/bitstream/2438/6706/2/Fultext.pdf

Treanor, Mike, et al. "AI-Based Game Design Patterns." University Georgia Institute Of Technology. 2015. Web. 11 Nov. 2016. http://www.cc.gatech.edu/~azook3/paper/treanor-fdg-2015.pdf

Lars Vogel. "Quick Links." Reading and Writing Files in Java (Input/Output) - Tutorial. Vogella GmbH, 27 Sept. 2016. Web. 28 Nov. 2016.

Appendix A

The probability tree code was adapted from Paliath's code for generic n-ary trees, adding the probabilities and selection processes as follows.

ProbabilityTree.java

```
package storygenerator;
import java.util.List;
import java.util.LinkedList;
import java.util.Random;

/**
 * @author Lindsey Harris
 */

class ProbabilitySelector<T> // Straightforward Probability Selection
{
    double sum;
    Random randomGenerator;

    public ProbabilitySelector()
    {
        super();
        randomGenerator = new Random();
        sum = 0.0;
    }

    public ProbabilityNode<T> selectOption(ProbabilityNode<T> choice)
    {
        if(!choice.hasChildren())
        {
            ProbabilityNode<T> nullNode = new ProbabilityNode();
            return nullNode;
        }
        choice.normalize();

        sum = choice.getChildAt(0).getProbability();
         // Get number in range 0 - 999
        int randomInt = randomGenerator.nextInt(1000);
        if(randomInt <= sum*1000)
        {
            return choice.getChildAt(0);
        }
        else
        {
            int i = 0;
            sum = 0.0;
            while (((sum * 1000) < randomInt) &&
                    (i<choice.getNumChildren()))
            {
                sum += choice.getChildAt(i).getProbability();
                i++;
            }
            return choice.getChildAt(i-1);
        }
    }
}


class AdvProbabilitySelector<T> // One-Step Memory
{
    double sum;
    int upperBound;
    int lowerBound;
    Random randomGenerator;

    public AdvProbabilitySelector()
    {
        super();
        this.randomGenerator = new Random();
        this.sum = 0.0;
        this.upperBound = 1;
        this.lowerBound = 0;
    }

    public ProbabilityNode<T> selectOption(ProbabilityNode<T> choice)
    {
        // Make sure choice has options to choose from
        if(!choice.hasChildren())
        {
            ProbabilityNode<T> nullNode = new ProbabilityNode();
            return nullNode;
        }
        //Make sure choice probabilities are distributed properly
        choice.normalize();

        // Initialize randomInt and sum
        int randomInt = randomGenerator.nextInt(1000);
        sum = choice.getChildAt(0).getProbability();
        ProbabilityNode<T> selectedNode;
        if(randomInt <= sum*1000) {
            selectedNode = choice.getChildAt(0);
        }
        else
        {
            int i = 0;
            sum = 0.0;
            while (((sum * 1000) < randomInt) &&
                    (i < choice.getNumChildren()))
            {
                sum += choice.getChildAt(i).getProbability();
                i++;
            }
            if(i>0){ selectedNode = choice.getChildAt(i-1); }
            else   { selectedNode = choice.getChildAt(i);}
        }

        double lb = selectedNode.lowerBound;
        double ub = selectedNode.upperBound;
        selectedNode.setProbability(selectedNode.getProbability()+lb);

       if(selectedNode.getProbability() < 0)
        {
            selectedNode.setProbability(0.0);
        }

       if(lb < 0)
        {
            lb = 0.0;
        }



        for(int i = 0; i < choice.getNumChildren(); i++)
        {
            ProbabilityNode<T> temp = choice.getChildAt(i);
            if(temp.getProbability() > ub)
            {
           choice.getChildAt(i).setProbability(ub);
            }
            double newProb = temp.getProbability()-lb;
            if(newProb < 0.0){newProb = 0.0;}
            choice.getChildAt(i).setProbability(newProb);
        }

        choice.normalize();

        return selectedNode;
    }
}
```

```
class ProbabilityNode<T>
{
    public T nodeObject;
    double probability;
    double upperBound;
    double lowerBound;
    public List<ProbabilityNode<T>> children;

    public ProbabilityNode()
    {
        this.children = new LinkedList<ProbabilityNode<T>>();
        this.upperBound = 1.0;
        this.lowerBound = 0.0;
    }

    public ProbabilityNode(T object, double prob)
    {
        this.children = new LinkedList<ProbabilityNode<T>>();
        this.nodeObject = object;
        this.probability = prob;
        this.upperBound = 1.0;
        this.lowerBound = 0.0;
    }

    public ProbabilityNode(T object,
                           double prob,
                           LinkedList<ProbabilityNode<T>> childList
                          )
    {
        this.nodeObject = object;
        this.probability = prob;
        this.children = childList;
        this.upperBound = 1.0;
        this.lowerBound = 0.0;
    }

     public ProbabilityNode(T object, double prob,
                           double lowBnd, double upBnd)
    {
        this.children = new LinkedList<ProbabilityNode<T>>();
        this.nodeObject = object;
        this.probability = prob;
        this.upperBound = upBnd;
        this.lowerBound = lowBnd;
    }
```

```
public ProbabilityNode(T object,
                          double prob,
                          LinkedList<ProbabilityNode<T>> childList,
                          double upBnd,
                          double lowBnd
                         )
    {
        this.nodeObject = object;
        this.probability = prob;
        this.children = childList;
        this.upperBound = upBnd;
        this.lowerBound = lowBnd;
    }

    public void setNodeObject(T object)
    {
        this.nodeObject = object;
    }

    public T getNodeObject()
    {
        return this.nodeObject;
    }

    public void setProbability(double prob)
    {
        this.probability = prob;
    }

    public double getProbability()
    {
        return this.probability;
    }

    public List<ProbabilityNode<T>> getChildren()
    {
        return children;
    }

    public void setChildren(List<ProbabilityNode<T>> children)
    {
        this.children = children;
    }

    public void addChild(ProbabilityNode<T> childNode)
    {
        this.children.add(childNode);
    }
```

```
public void addChildAt(int index, ProbabilityNode<T> child)
            throws IndexOutOfBoundsException
    {
        this.children.add(index, child);
    }

    public ProbabilityNode<T> getChildAt(int index)
            throws IndexOutOfBoundsException
    {
        return this.children.get(index);
    }

    public void removeChildAt(int index)
            throws IndexOutOfBoundsException
    {
        this.children.remove(index);
    }

    public int getNumChildren()
    {
        return this.children.size();
    }

    public boolean hasChildren()
    {
        return (getNumChildren() > 0);
    }

    public void normalize()
    {
        double newProbSum = 0;

         for(int i = 0; i < this.getNumChildren(); i++)
         {
            newProbSum += this.getChildAt(i).getProbability();
         }
         for(int i = 0; i < this.getNumChildren(); i++)
         {
            double prob = this.getChildAt(i).getProbability();
            this.getChildAt(i).setProbability(prob / newProbSum);
         }
    }
}
```

```
public class ProbabilityTree<T>
{
    ProbabilityNode<T> root;
    int iterator;
    AdvProbabilitySelector<T> selector;

    public ProbabilityTree()
    {
        super();
        this.iterator = 0;
        this.selector = new AdvProbabilitySelector();
    }

    public ProbabilityTree(ProbabilityNode<T> rootNode)
    {
        this.root = rootNode;
        this.iterator = 0;
        this.selector = new AdvProbabilitySelector();
    }

    public ProbabilityNode<T> selectNodeChild(ProbabilityNode<T> node)
    {
        return selector.selectOption(node);
    }
}
```

Appendix B

The implementation for the Story Generator Application is comprised of two files: StoryGenerator.java and GUIWindow.java. The code for these files is as follows.

StoryGenerator.java

```
package storygenerator;
import StoryGeneratorGUI.GUIWindow;
/**
 *
 * @author Lindsey Harris
 */
public class StoryGenerator {
    public static String storyText;
    public static ProbabilityTree<Scene> sceneMap;
    Character protagonist;

    public StoryGenerator() {
        storyText = "";
    }

    public static void main(String[] args) {

        sceneMap = setupDemo();
        GUIWindow gui = new GUIWindow();
    }

    public static String getText()
    {
        return storyText;
    }

    public static void generateStory(ProbabilityTree<Scene> tree)
    {
        sceneMap = setupDemo();
        ProbabilityNode<Scene> temp = tree.root;
        storyText = temp.nodeObject.sceneText;
        while(temp.hasChildren())
        {
            temp = tree.selectNodeChild(temp);
            System.out.print(temp.nodeObject.sceneName+"\n");
            storyText = storyText+temp.nodeObject.sceneText;
        }
        System.out.print(storyText);
    }
    public static ProbabilityTree<Scene> setupDemo() {
        Scene rootScene = new Scene("Talk to King",
                "     Once upon a time, in a faraway kingdom, there
was a sickness in the land. The king’s sister, who\n"+
                "was loved by all, was one of the many who became ill,
and much to the sorrow of the kingdom, she\n"+
                "passed away. She had a son, whom the king adored, but
who had not yet begun working on a trade,\n"+
                "so the king decided to take him under his wing.\n\n"+
                "The king summoned him into the throne room and told
him:\n\n"+
                "     ‘You are growing older, and have still not
decided what you are going to do with your life. It is time\n"+
                "     you learned a trade of some sort, and here,
today, I have summoned the Captain of the Guard and a\n"+
                "     talented Minstrel of the King’s court for you to
talk to. See if you would like to study under one of them.\n"+
                "     If so, you can become something great, but if
you do not find something to do here, you will have to\n"+
                "     leave and make a living elsewhere, so let me
know your decision by the end of the day.’\n\n"+
                "And with that the king dismissed him from his
presence.\n\n"
                );

        Scene talkMinstrelScene1 = new Scene("Talk to Minstrel",
                "     The boy went to the Minstrel, who showed him a
flute and a lyre. He told him about what a minstrel’s\n"+
                "life was like-- attending parties and events and
providing entertainment for the king. He offered to teach\n"+
                "him the lyre.\n\n");
        Scene talkMinstrelScene2 = new Scene("Talk to Minstrel",
                "     The boy went to the Minstrel, who showed him a
song on the lyre.\n\n");

        Scene talkKnightScene2 = new Scene("Talk to Knight",
                "     The boy went to the Captain of the Guard, who
demonstrated some of the basic training techniques, telling \n"+
                "him that if he trained hard, he could join the forces
to fight for the king.\n\n");

        Scene talkKnightScene1 = new Scene("Talk to Knight",
                "     The boy went to the Captain of the Guard, who
showed him his sword and shield. He demonstrated some of\n"+
                "the basic training techniques, and spoke of the glory
that came with being a knight and defending the kingdom.\n"+
                "Then he told him if he trained hard, he could join
the forces to fight for the king.\n\n");
```

```
        Scene talkPrincessScene = new Scene("Talk to Princess",
                "The boy noticed the beautiful princess visiting the
king's court, so he went to talk to her. She had a lovely\n"+
                "disposition and seemed happy to talk to him.\n\n");

        Scene playInstrumentScene = new Scene("Play Instrument",
                "The boy thought the lyre looked interesting, and so
asked if he could try it. The minstrel handed it over,\n"+
                "and showed him how to play a song. He said,\n\n"+
                "     ‘If you keep practicing, then soon, you should
be able to perform with me at some of the king’s events.’\n\n");

        Scene chooseMinstrelScene = new Scene("Choose Minstrel",
                "     After thinking about it for a little while, the
boy decided that the life of a minstrel sounded like the life he\n"+
                "would like to have, so he went to the king and told
him he would like to be a minstrel. The king\n"+
                "congratulated him on his decision and welcomed him
into his court.\n\nThe End.\n\n");
        Scene trainScene = new Scene("Train",
                "The boy decided to try some of the Captain's training
techniques, and so spent some time practicing his combat
skills.\n\n");

        Scene chooseKnightScene = new Scene("Choose Knight",
                "     After thinking about it for a while, the boy
decided that defending the kingdom was a task he would be proud to\n"+
                "spend his life doing, so he went to the king and told
him,\n\n"+
                "     'If it pleases your majesty, I would like to
study under the Captain of the Guard to become a knight of the
king's\n"+
                "      court.'\n\n"+
                "This pleased the king very much, and so he
congratulated him on his decision and welcomed him into his
guard.\n\nThe End.\n\n");

        Scene romanceScene = new Scene("Romance Princess",
                "They went walking and talked for a while. He
complemented her frequently, speaking to her quite fondly.\n\n");

        Scene proposeScene = new Scene("Propose to Princess",
                "     The young man was quite happy with the princess,
and thought of her highly. After much consideration, he asked her what
she\n"+
                "thought of marriage. He told her that if she was
pleased with him, he could not think of anyone that he would rather
marry.\n"+
                "Flattered, she thought for a moment, and pleased with
the proposition, accepted his proposal--under condition that it
pleased the\n"+
                "king. So the young man went to the king, and told him
that he would like to settle down and that if it pleased the king, he
wished\n"+
                "to marry the beautiful princess who was visiting the
king's court. The king was pleased at this news, and congratulated
them both.\n"+
                "So the two were married, and the king gave them land,
making the young man a lord over a large area of the kingdom, where
the two\n"+
                "of them lived happily for many years.\n\nThe
End.\n\n");

        //Root
        ProbabilityNode<Scene> rootNode  =
                new ProbabilityNode(rootScene, 1);

        //Level 1
        ProbabilityNode<Scene> minstrelNode1  =
                new ProbabilityNode(talkMinstrelScene1,0.3333333);
        ProbabilityNode<Scene> knightNode1  =
                new ProbabilityNode(talkKnightScene1,    0.3333333);
        ProbabilityNode<Scene> princessNode1 =
                new ProbabilityNode(talkPrincessScene, 0.3333333);

        //Level 2
        ProbabilityNode<Scene> minstrelNode2  =
                new ProbabilityNode(talkMinstrelScene1, 0.25, 0.05,
                                   1.0);
        ProbabilityNode<Scene> knightNode2  =
                new ProbabilityNode(talkKnightScene1, 0.25, 0.05,
                                   1.0);
        ProbabilityNode<Scene> princessNode2 =
                new ProbabilityNode(talkPrincessScene, 0.25, 0.05,
                                   1.0);
        ProbabilityNode<Scene> playNode  =
                new ProbabilityNode(playInstrumentScene, 0.25, 0.05,
                                   1.0);
        ProbabilityNode<Scene> chooseMinstrelNode =
                new ProbabilityNode(chooseMinstrelScene, 0.25, 0.05,
                                   1.0);
        ProbabilityNode<Scene> trainNode =
                new ProbabilityNode(trainScene, 0.25, 0.05, 1.0);
        ProbabilityNode<Scene> chooseKnightNode =
                new ProbabilityNode(chooseKnightScene, 0.25, 0.05,
                                   1.0);
        ProbabilityNode<Scene> romanceNode =
                new ProbabilityNode(romanceScene, 0.25, 0.05, 1.0);
        ProbabilityNode<Scene> proposeNode =
                new ProbabilityNode(proposeScene, 0.25, 0.05, 1.0);
```

```
//Build Tree from the bottom up
// Level 4
minstrelNode2.addChild(playNode);
minstrelNode2.addChild(knightNode2);
minstrelNode2.addChild(princessNode2);
minstrelNode2.addChild(chooseMinstrelNode);

knightNode2.addChild(trainNode);
knightNode2.addChild(minstrelNode2);
knightNode2.addChild(princessNode2);
knightNode2.addChild(chooseKnightNode);

princessNode2.addChild(romanceNode);
princessNode2.addChild(knightNode2);
princessNode2.addChild(minstrelNode2);
princessNode2.addChild(proposeNode);

// Level 3
playNode.addChild(knightNode2);
playNode.addChild(princessNode2);
playNode.addChild(chooseMinstrelNode);
playNode.addChild(playNode); //Add recursive element

trainNode.addChild(princessNode2);
trainNode.addChild(minstrelNode2);
trainNode.addChild(chooseKnightNode);
trainNode.addChild(trainNode); //Add recursive element

romanceNode.addChild(minstrelNode2);
romanceNode.addChild(knightNode2);
romanceNode.addChild(proposeNode);
romanceNode.addChild(romanceNode); //Add recursive element

// Level 2
minstrelNode1.addChild(playNode);
minstrelNode1.addChild(knightNode2);
minstrelNode1.addChild(princessNode2);
minstrelNode1.addChild(chooseMinstrelNode);

knightNode1.addChild(trainNode);
knightNode1.addChild(minstrelNode2);
knightNode1.addChild(princessNode2);
knightNode1.addChild(chooseKnightNode);

princessNode1.addChild(romanceNode);
princessNode1.addChild(knightNode2);
princessNode1.addChild(minstrelNode2);
princessNode1.addChild(proposeNode);
```

```
        // Level 1
        rootNode.addChild(minstrelNode1);
        rootNode.addChild(knightNode1);
        rootNode.addChild(princessNode1);

        ProbabilityTree<Scene> demoTree =
                new ProbabilityTree(rootNode);
        return demoTree;
    }
}
```

Appendix C

Some of the sample stories created by the story generator are listed below. They are not well-written as stories, but they illustrate how the story generator works with the demo tree. There are some straightforward stories as well as some more complicated ones to show some of the diversity that the generator created.

Straightforward Stories

The following stories never selected a repeating option.

Marrying the Princess - 2

Once upon a time, in a faraway kingdom, there was a sickness in the land. The king’s sister, who was loved by all, was one of the many who became ill, and much to the sorrow of the kingdom, she passed away. She had a son, whom the king adored, but who had not yet begun working on a trade, so the king decided to take him under his wing. The king summoned him into the throne room and told him:

‘You are growing older, and have still not decided what you are going to do with your life. It is time you learned a trade of some sort, and here, today, I have summoned the Captain of the Guard and a talented Minstrel of the King’s court for you to talk to. See if you would like to study under one of them. If so, you can become something great, but if you do not find something to do here, you will have to leave and make a living elsewhere, so let me know your decision by the end of the day.’

And with that the king dismissed him from his presence.

The boy noticed the beautiful princess visiting the king's court, so he went to talk to her. She had a lovely disposition and seemed happy to talk to him.

The young man was quite happy with the princess, and thought of her highly. After much consideration, he asked her what she thought of marriage. He told her that if she was pleased with him, he could not think of anyone that he would rather marry. Flattered, she thought for a moment, and pleased with the proposition, accepted his proposal--under condition that it pleased the king. So the young man went to the king, and told him that he would like to settle down and that if it pleased the king, he wished to marry the beautiful princess who was visiting the king's court. The king was pleased at this news, and congratulated them both. So the two were married, and the king gave them land, making the young man a lord over a large area of the kingdom, where the two of them lived happily for many years.

The End.

Becoming a Minstrel - 3

Once upon a time, in a faraway kingdom, there was a sickness in the land. The king’s sister, who was loved by all, was one of the many who became ill, and much to the sorrow of the kingdom, she passed away. She had a son, whom the king adored, but who had not yet begun working on a trade, so the king decided to take him under his wing. The king summoned him into the throne room and told him:

‘You are growing older, and have still not decided what you are going to do with your life. It is time you learned a trade of some sort, and here, today, I have summoned the Captain of the Guard and a talented Minstrel of the King’s court for you to talk to. See if you would like to study under one of them. If so, you can become something great, but if

you do not find something to do here, you will have to leave and make a living elsewhere, so let me know your decision by the end of the day.'
And with that the king dismissed him from his presence.

The boy went to the Minstrel, who showed him a flute and a lyre. He told him about what a minstrel's life was like-- attending parties and events and providing entertainment for the king. He offered to teach him the lyre.

After thinking about it for a little while, the boy decided that the life of a minstrel sounded like the life he would like to have, so he went to the king and told him he would like to be a minstrel. The king congratulated him on his decision and welcomed him into his court.
The End.

Becoming a Knight - 32

Once upon a time, in a faraway kingdom, there was a sickness in the land. The king's sister, who was loved by all, was one of the many who became ill, and much to the sorrow of the kingdom, she passed away. She had a son, whom the king adored, but who had not yet begun working on a trade, so the king decided to take him under his wing. The king summoned him into the throne room and told him:

'You are growing older, and have still not decided what you are going to do with your life. It is time you learned a trade of some sort, and here, today, I have summoned the Captain of the Guard and a talented Minstrel of the King's court for you to talk to. See if you would like to study under one of them. If so, you can become something great, but if you do not find something to do here, you will have to leave and make a living elsewhere, so let me know your decision by the end of the day.'

And with that the king dismissed him from his presence.

The boy went to the Captain of the Guard, who showed him his sword and shield. He demonstrated some of the basic training techniques, and spoke of the glory that came with being a knight and defending the kingdom. Then he told him if he trained hard, he could join the forces to fight for the king.

After thinking about it for a while, the boy decided that defending the kingdom was a task he would be proud to spend his life doing, so he went to the king and told him,

'If it pleases your majesty, I would like to study under the Captain of the Guard to become a knight of the king's court.'

This pleased the king very much, and so he congratulated him on his decision and welcomed him into his guard.

The End.

More Complicated Examples

The following stories demonstrate more of the decision-making process of the AI with one-step memory. Some of the examples are repetitive, mimicking a desire to follow a specific path, while others mimic an indecisive character that makes up his mind at the last minute.

Marrying the Princess - 26

Once upon a time, in a faraway kingdom, there was a sickness in the land. The king’s sister, who was loved by all, was one of the many who became ill, and much to the sorrow of the kingdom, she passed away. She had a son, whom the king adored, but who had not yet begun working on a trade, so the king decided to take him under his wing. The king summoned him into the throne room and told him:

‘You are growing older, and have still not decided what you are going to do with your life. It is time you learned a trade of some sort, and here, today, I have summoned the Captain of the Guard and a talented Minstrel of the King’s court for you to talk to. See if you would like to study under one of them. If so, you can become something great, but if you do not find something to do here, you will have to leave and make a living elsewhere, so let me know your decision by the end of the day.’

And with that the king dismissed him from his presence.

The boy went to the Captain of the Guard, who showed him his sword and shield. He demonstrated some of the basic training techniques, and spoke of the glory that came with being a knight and defending the kingdom. Then he told him if he trained hard, he could join the forces to fight for the king.

The boy noticed the beautiful princess visiting the king's court, so he went to talk to her. She had a lovely disposition and seemed happy to talk to him.

The boy went to the Minstrel, who showed him a flute and a lyre. He told him about what a minstrel’s life was like-- attending parties and events and providing entertainment for the king. He offered to teach him the lyre.

The boy noticed the beautiful princess visiting the king's court, so he went to talk to her. She had a lovely disposition and seemed happy to talk to him.

They went walking and talked for a while. He complemented her frequently, speaking to her quite fondly.

The boy went to the Minstrel, who showed him a flute and a lyre. He told him about what a minstrel’s life was like-- attending parties and events and providing entertainment for the king. He offered to teach him the lyre.

The boy thought the lyre looked interesting, and so asked if he could try it. The minstrel handed it over, and showed him how to play a song. He said,

‘If you keep practicing, then soon, you should be able to perform with me at some of the king’s events.’

The boy noticed the beautiful princess visiting the king's court, so he went to talk to her. She had a lovely disposition and seemed happy to talk to him.

The boy went to the Minstrel, who showed him a flute and a lyre. He told him about what a minstrel’s life was like-- attending parties and events and providing entertainment for the king. He offered to teach him the lyre.

The boy thought the lyre looked interesting, and so asked if he could try it. The minstrel handed it over, and showed him how to play a song. He said,

‘If you keep practicing, then soon, you should be able to perform with me at some of the king’s events.’

The boy noticed the beautiful princess visiting the king's court, so he went to talk to her. She had a lovely disposition and seemed happy to talk to him.

The young man was quite happy with the princess, and thought of her highly. After much consideration, he asked her what she thought of marriage. He told her that if she was pleased with him, he could not think of anyone that he would rather marry. Flattered, she thought for a moment, and pleased with the proposition, accepted his proposal--under condition that it pleased the king. So the young man went to the king, and told him that he would like to settle down and that if it pleased the king, he wished to marry the beautiful princess who was visiting the king's court. The king was pleased at this news, and congratulated them both. So the two were married, and the king gave them land, making the young man a lord over a large area of the kingdom, where the two of them lived happily for many years.

The End.

Becoming a Minstrel - 30

Once upon a time, in a faraway kingdom, there was a sickness in the land. The king’s sister, who was loved by all, was one of the many who became ill, and much to the sorrow of the kingdom, she passed away. She had a son, whom the king adored, but who had not yet begun working on a trade, so the king decided to take him under his wing. The king summoned him into the throne room and told him:

‘You are growing older, and have still not decided what you are going to do with your life. It is time you learned a trade of some sort, and here, today, I have summoned the Captain of the Guard and a talented Minstrel of the King’s court for you to talk to. See if you would like to study under one of them. If so, you can become something great, but if you do not find something to do here, you will have to leave and make a living elsewhere, so let me know your decision by the end of the day.’

And with that the king dismissed him from his presence.

The boy went to the Captain of the Guard, who showed him his sword and shield. He demonstrated some of the basic training techniques, and spoke of the glory that came with being a knight and defending the kingdom. Then he told him if he trained hard, he could join the forces to fight for the king.

The boy decided to try some of the Captain's training techniques, and so spent some time practicing his combat skills.

The boy decided to try some of the Captain's training techniques, and so spent some time practicing his combat skills.

The boy decided to try some of the Captain's training techniques, and so spent some time practicing his combat skills.

The boy went to the Minstrel, who showed him a flute and a lyre. He told him about what a minstrel’s life was like-- attending parties and events and providing entertainment for the king. He offered to teach him the lyre.

The boy noticed the beautiful princess visiting the king's court, so he went to talk to her. She had a lovely disposition and seemed happy to talk to him.

The boy went to the Minstrel, who showed him a flute and a lyre. He told him about what a minstrel’s life was like-- attending parties and events and providing entertainment for the king. He offered to teach him the lyre.

The boy thought the lyre looked interesting, and so asked if he could try it. The minstrel handed it over, and showed him how to play a song. He said,

‘If you keep practicing, then soon, you should be able to perform with me at some of the king’s events.’

The boy thought the lyre looked interesting, and so asked if he could try it. The minstrel handed it over, and showed him how to play a song. He said,

‘If you keep practicing, then soon, you should be able to perform with me at some of the king’s events.’

The boy noticed the beautiful princess visiting the king's court, so he went to talk to her. She had a lovely disposition and seemed happy to talk to him.

The boy went to the Minstrel, who showed him a flute and a lyre. He told him about what a minstrel’s life was like-- attending parties and events and providing entertainment for the king. He offered to teach him the lyre.

The boy thought the lyre looked interesting, and so asked if he could try it. The minstrel handed it over, and showed him how to play a song. He said,

‘If you keep practicing, then soon, you should be able to perform with me at some of the king’s events.’

After thinking about it for a little while, the boy decided that the life of a minstrel sounded like the life he would like to have, so he went to the king and told him he would like to be a minstrel. The king congratulated him on his decision and welcomed him into his court.

The End.

Becoming a Knight - 22

Once upon a time, in a faraway kingdom, there was a sickness in the land. The king’s sister, who was loved by all, was one of the many who became ill, and much to the sorrow of the kingdom, she passed away. She had a son, whom the king adored, but who had not yet begun working on a trade, so the king decided to take him under his wing. The king summoned him into the throne room and told him:

‘You are growing older, and have still not decided what you are going to do with your life. It is time you learned a trade of some sort, and here, today, I have summoned the Captain of the Guard and a talented Minstrel of the King’s court for you to talk to. See if you would like to study under one of them. If so, you can become something great, but if you do not find something to do here, you will have to leave and make a living elsewhere, so let me know your decision by the end of the day.’

And with that the king dismissed him from his presence.

The boy went to the Captain of the Guard, who showed him his sword and shield. He demonstrated some of the basic training techniques, and spoke of the glory that came with being a knight and defending the kingdom. Then he told him if he trained hard, he could join the forces to fight for the king.

The boy decided to try some of the Captain's training techniques, and so spent some time practicing his combat skills.

The boy decided to try some of the Captain's training techniques, and so spent some time practicing his combat skills.

The boy went to the Minstrel, who showed him a flute and a lyre. He told him about what a minstrel’s life was like-- attending parties and events and providing entertainment for the king. He offered to teach him the lyre.

The boy went to the Captain of the Guard, who showed him his sword and shield. He demonstrated some of the basic training techniques, and spoke of the glory that came with being a knight and defending the kingdom. Then he told him if he trained hard, he could join the forces to fight for the king.

The boy decided to try some of the Captain's training techniques, and so spent some time practicing his combat skills.

After thinking about it for a while, the boy decided that defending the kingdom was a task he would be proud to spend his life doing, so he went to the king and told him,

'If it pleases your majesty, I would like to study under the Captain of the Guard to become a knight of the king's court.'

This pleased the king very much, and so he congratulated him on his decision and welcomed him into his guard.

The End.