Jonathan Louie
CS7637

Assignment 3

INTRO

The Odd One Out problem scheme was designed as a test for human intelligence and cognition. It features 9 figures, each with varying attributes such as shape, shape count, color, size, and formation, and the subject must choose the single figure that least fits the group. Despite these problems primarily being used to test humans, an artificial intelligence agent can be developed to solve them in a human-like manner by implementing ideas of classification and logic.

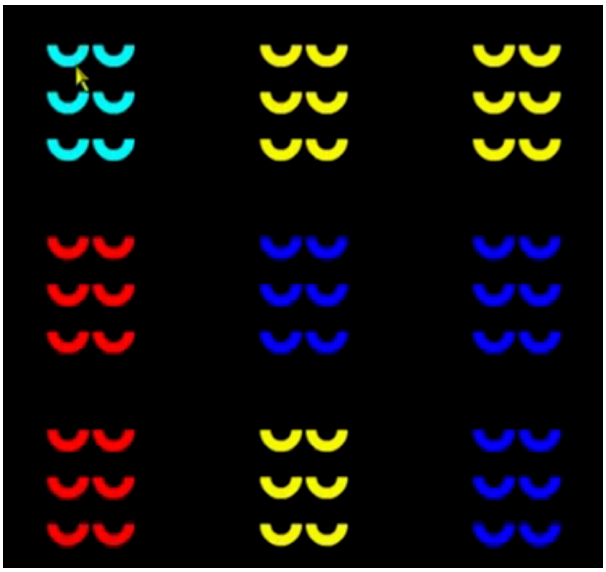

*Figure 1. An example of an Odd One Out problem- the answer would be the top left figure.*

THE APPROACH

The artificial intelligence agent would use classification and logic to solve the Odd One Out problems. These are two concepts commonly used by humans on the day-to-day and always used when solving problems such as the Odd One Out. By implementing these concepts with the agent, it will give the agent a more human-like problem-solving process.

Classification is the process of mapping sets of percepts into single concepts. In the case of these problems, attributes of each figure are the percepts used to form the concept. By observing all features of each figure, a target concept figure can be formed. This concept figure would contain all of the attributes necessary to be considered “in the group” and the single figure not matching this concept would be labeled as the “odd one out”.

Logic has two parts, the knowledge base and the rules of inference. Knowledge about the figures in the problems would be gathered by looking at the attributes of each

figure. Based off of this knowledge, the agent can use predicates (functions used to map attributes to Boolean values) to infer what is and isn't true about each figure.

THE AGENT DESIGN

The agent would use a multi-step approach that combines the strengths of both logic and classification. The first step would be to build a knowledge base about the problem by gathering as much information as possible about even individual figure. After that, each figure would be run through predicates that would determine even more information about each figure and the problem as a whole. Lastly, with all of the total information gathered, the single figure not matching a common concept would be selected as the odd one out.

Preliminary information gathering would place each figure's information into an object. Each figure would be a dictionary where the key/value pairs would be represented by attribute-name/attribute-value. A sample figure would be stored like this:

| ATTRIBUTE NAME | ATTRIBUTE VALUE |
|---|---|
| Shape | Half-circle |
| Shape count | 6 |
| Formation | Rectangle |
| Color | Cyan |
| Position | Top-left |

Predicates would be able to give the agent further information by answering true/false questions that the agent might have such as:

- Is this figure identical to another?
- Is this figure's attribute value unique?
- Is this figure's attribute1 value/attribute2 value a unique pair?

The questions would be answered for each figure and for many attributes.

Using the results from the information gathered before, each figure in the problem would undergo a series of evaluations to determine which one is the most unique. The pseudocode is as follows:

IF figure is only one without an identical match, select as Odd One Out
ELSE IF figure has most unique attribute values for select as Odd One Out
ELSE IF figure has most attributes values from list of least common attribute values, select as Odd One Out
ELSE IF do not choose any figure as an Odd One Out

THE CHALLENGES

Solving Odd One Out problems uses reasoning and pattern-finding skills, both of which are commonly used by humans often enough to the point of taking them for granted. Because of this, recreating these processes in an application is a difficult task. A large number of comparisons is necessary to thoroughly analyze each problem to find

what is required for a figure to be considered the Odd One Out. The criterion for this changes from problem to problem, as does overall difficulty. The agent must be able to solve the easier problems quickly, while still being flexible enough to handle the more challenging ones.