KBAI for Raven's Progressive Matrices: Final Report

Luke Pratt
lpratt30@gatech.edu

***Abstract*—Raven's Progressive Matrices, being a measure of generalized human intelligence, make an excellent testing ground for AI techniques. In this paper, techniques of knowledge based artificial intelligence are applied to resolve RPM questions across simplified subsets of the test alongside 48 of the original questions. Primarily by using generate-and-test with means-ends-analysis, the Agent was able to achieve 75/96 correct answers across the mock B/C/D/E sets and 23/48 of the provided real RPM questions.**

1.1 INTRODUCTION

Knowledge Based Artificial Intelligence (KBAI) is a subset of AI that focuses on using *knowledge* about a problem to reach a solution using a range of decision-making strategies, many of the strategies similar to those of humans. Knowledge in KBAI is as we the programmer define it; it can be any useful facet of a problem that can be collected from the problem. KBAI concepts are valuable because of how broadly applicable and innately intuitive they are. In essence, KBAI is a way for a programmer to translate the way they solve a problem into instructions for a computer to replicate.

Raven's Progressive Matrices (RPM) are a form of testing for generalized human intelligence where knowledge learned from previous problems across a test is accumulated to determine answers to more difficult questions (Leavitt V.M, *Raven Progressive Matrices*). As it tests for generalized human intelligence and requires the use of knowledge for solving problems, it makes an excellent test of KBAI. While learning is in the sphere of KBAI, in this paper the focus is on having the problem-solving methods infused into it from the author.

1.2 Agent functionality- transform method

Consider the below image in figure 1 of a set B problem:

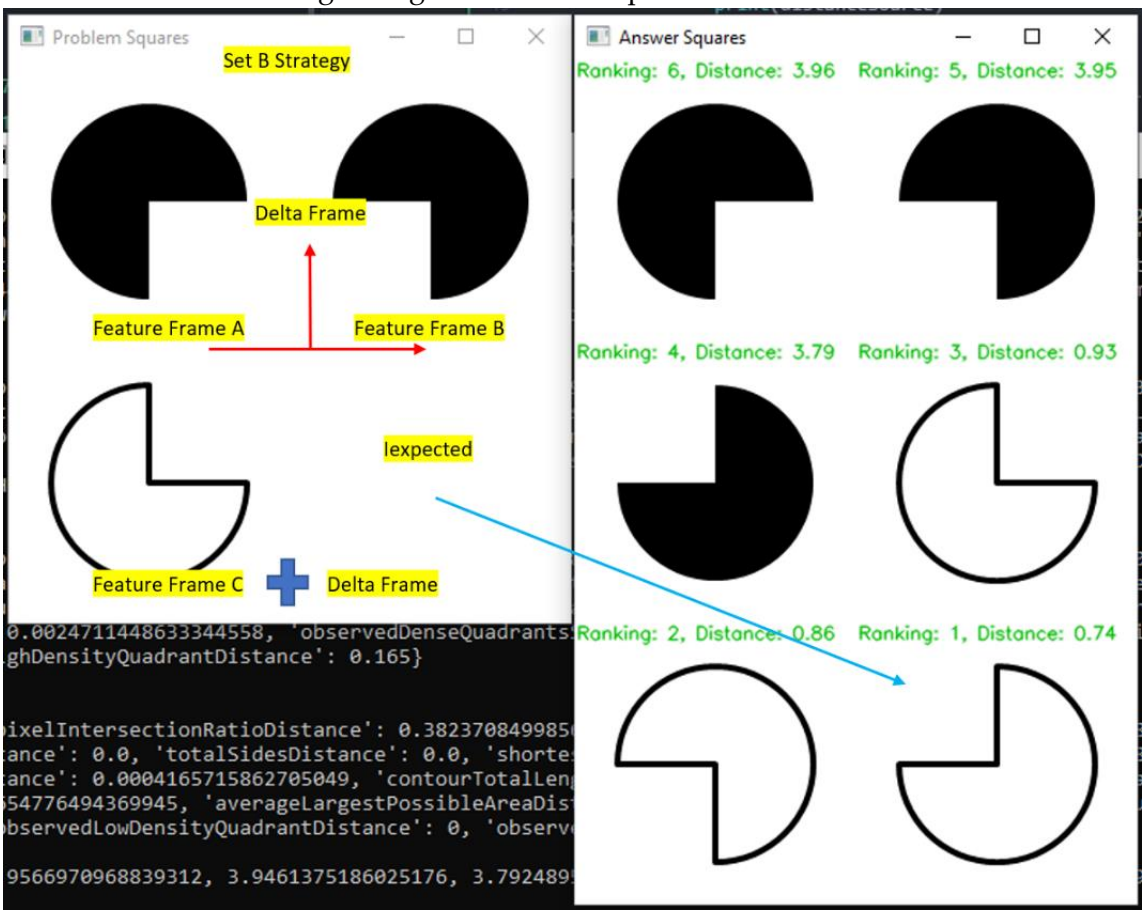

Figure 1- *Set B Strategy*

The Agent first performs feature extraction for each image in the problem and represents each image's features in a *frame*. Each frame defines the knowledge the Agent has about the image. The features the Agent collects are summarized in the below table 1.

Next, the Agent characterizes the deltas between the A and B frames. These deltas describe a transformation that occurred. Then, the Agent applies the deltas to the C frame preceding the blank answer. The features of C plus the transformation between A and B **generates** what the Agent expects to be the answer image. Finally, the Agent scans the answer images and searches for the frame with the lowest **distance** to the expected frame. This method will be referred to as the transform method.

Feature	Knowledge
Black Pixel Ratio[2]	How full is the image
Total Sides of all Shapes in Image	What sorts of shapes are there
Contour Count	How many shapes are there
Maximum Euclidian Distance	How far away are the farthest things
Average Area	What is the average shapes size
Longest Contour Length	What is the largest shapes length
Shortest Contour Length	What is the smallest shapes length
Total Contour Length	What is the total shape length
Densest Black Pixel Quadrant	If non-symmetrical, what is the location

Table 1- *Summary of features*

This Agent doesn't have any understanding of the meaning, or *ontology*, of what it is looking at. For example, it makes no attempt to classify a contour of 3 sides to be a triangle. However, while the Agent won't know what a triangle is, it will know that there is a 3-sided shape inside of the image.

These feature are not so intuitive for a human, but they do a good job of holding the information the Agent needs. Additionally, by not using ontology, the Agent generalizes better to unknown problems.

All of these features are only important in relation to how they change or don't change across a problem. Shown below in table 2 is how the Agent uses deltas between features to calculate distances:

Feature	Distance	Meaning
Pixel Intersection Ratio[2]	$\lvert(\text{Expected} - \text{Observed})\rvert^{1/2}$	How much overlap do the images have?
Black Pixel Ratio[2]	$\lvert(\text{Expected} - \text{Observed})\rvert$	How much darker is one image?
Total Sides of all Shapes in Image	$(\lvert(\text{Expected} - \text{Observed})\rvert/\text{Observed})^{1/2}$	What is the difference in the types of shapes?
Contour Count	$(\lvert(\text{Expected} - \text{Observed})\rvert/\text{Observed})^{1/2}$	What is the difference in the number of shapes?
Maximum Euclidian Distance	$(\lvert(\text{Expected} - \text{Observed})\rvert/\text{Observed})^{1/2}$	Are things getting closer or further apart?
Average Area	$\lvert((\text{Expected} - \text{Observed})/\text{Observed})\rvert$	Are things getting bigger or smaller?
Longest Contour Length	$\lvert((\text{Expected} - \text{Observed})/\text{Observed})\rvert * .33$	Did the biggest thing change?
Shortest Contour Length	$\lvert((\text{Expected} - \text{Observed})/\text{Observed})\rvert * .33$	Did the smallest thing change?
Total Contour Length	$\lvert((\text{Expected} - \text{Observed})/\text{Observed})\rvert * .33$	How did all the things change overall?
Densest Black Pixel Quadrant	If(Expected != Observed) += .33/2	Where is the center of gravity moving to?

Table 2- *Distance Calculations*

The goal was for a feature to be broadly applicable and not overlap with other features. An ideal feature has prominent distance changes in some problems and becomes the defining feature to solve the problem with, while in other problems is totally neutral and has no changes. For example, in a problem above such as figure 1, there are no changes to the size of any shape, so the contour length distances will not play into generating the expected D frame. In fact, they will constrain the Agent to only generate contour lengths similar to the C frame, because the Agent previously saw no change to them. This means some of the best features are those that best describe the image while not necessarily being important for a transformation within a problem.

Each feature is normalized to a minimum of 0 and a maximum of 1.0 such that the square root increases the value and does so more for smaller values for detecting subtle changes of critical features. Note that the various contour lengths are multiplied by $1/3^{rd}$ to justify that they are overlapping features.

1.3 Agent functionality- legality method

Before selecting an answer, the Agent also *tests* whether or not an answer or a legal or illegal choice. This will be referred to as the legality method.

The legality method checks 3 rules. First, it determines if the number of black pixels at the end of each row are the same as the beginning of the row. If so, the Agent eliminates answers that don't keep this pattern. Alternatively, the Agent determines if the number of black pixels must always increase going from left to right in the problem. Finally, the Agent checks how many unique images are within the problem image by using a pixel intersection ratio with a threshold. If every single image is unique, the Agent eliminates answers that are within the problem. Shown in figure 2 is an example the Agent fails without using legality.

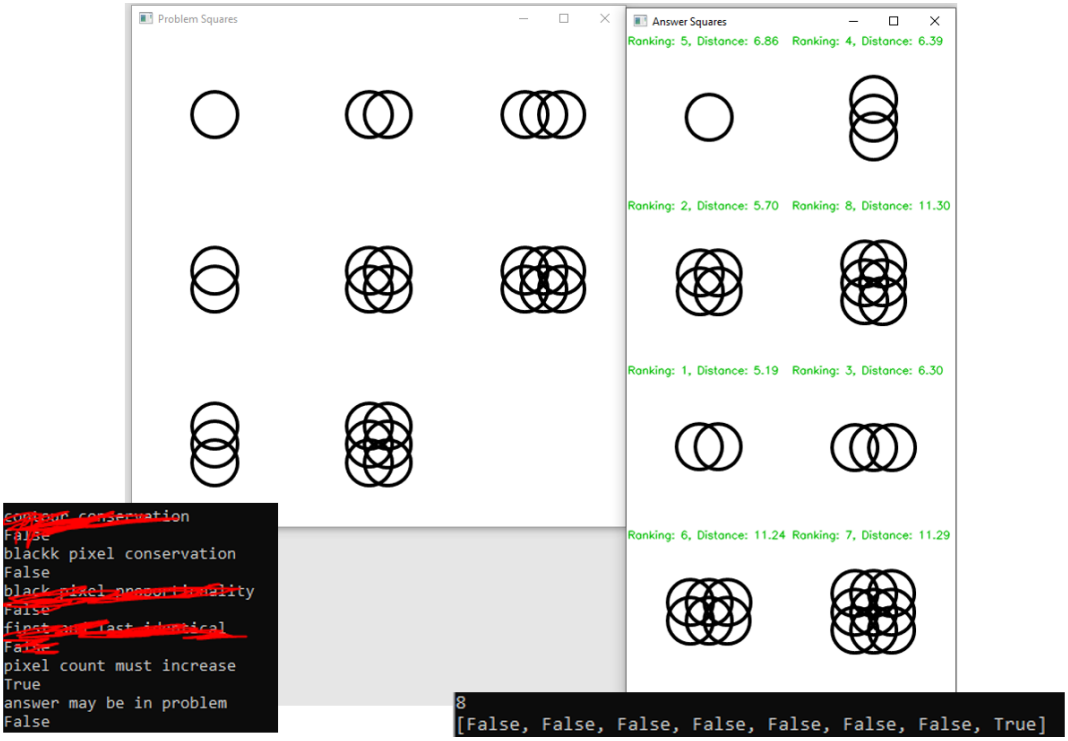

Figure 2- *Legality Method*

The main value is eliminating answers that are also in the problem. This increases performance as images in the problem tend to be similar to **Iexpected**. Secondly, the additional rules within the Agent's legality method help the Agent resolve fringe cases that its features and distances are not suited for. Unfortunately, across sets C and D, the Agent still struggles in problems that are more abstract, such as figure 2.

1.4 Agent functionality- set C

This Agent has variations in its approach dependent on the problem type (2x2 vs 3x3) or problem set B-E. However, aside from set E, the fundamental mechanics are similar for all sets. Although set C is a 3x3, the Agent still treats it as a 2x2. This is because set C is fundamentally the same as set B in terms of left-right transformations and isn't necessarily consistent across rows and columns.

Shown below in figure 3 is the Agent attempting to solve a set C problem:

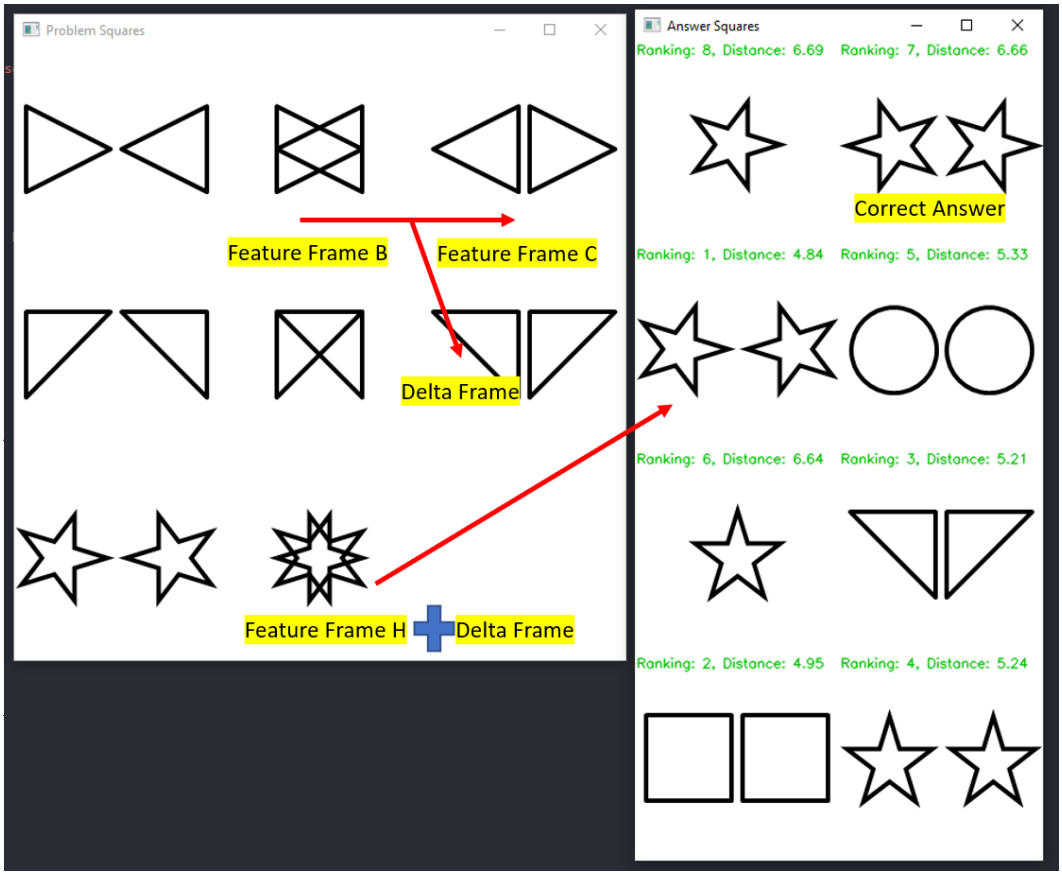

Figure 3- *Set C Solution*

The Agent fails to solve the above problem because the transformation between B and C is not well defined, nor is frame H well defined. The Agent's strongpoint in set C are problems that have sweeping changes in features between B and C, for features clearly defined by the Agent, between distinct entities.

1.5 Agent functionality- set D

Set D, although being slightly more abstract, gives hints due to consistency between rows. Because of this, the Agent is able to generate multiple **Iexpected** frames coming from multiple angles, as shown below in figure 4. Of the three generated Iexpected frames, the Agent selects that answer that has the single lowest distance to 1 of the 3 frames.

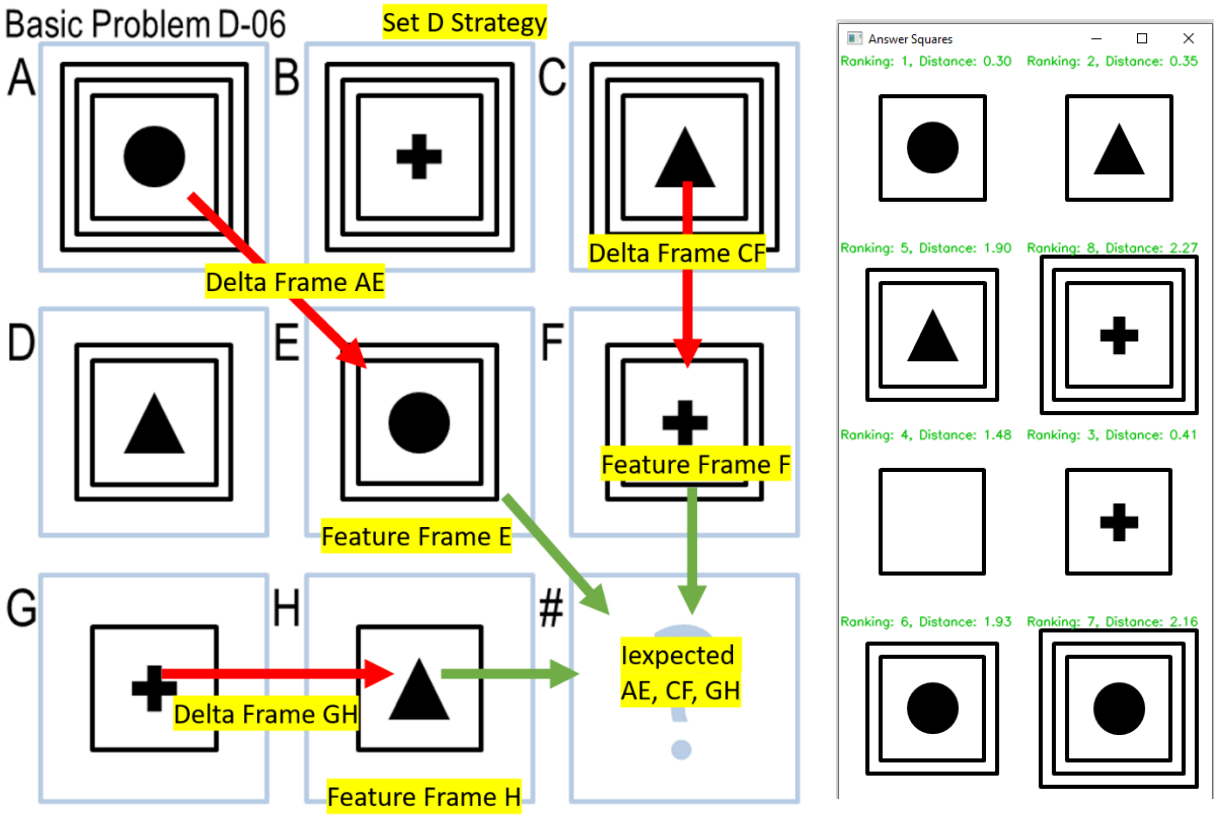

Figure 4- *Set D Solution*

This method breaks down when there is both a major conflict between any key feature between two otherwise similar transformations, and/or when a transformation occurs that isn't well defined by the Agent. For example, in the below figure 5, both issues happen at the same time.

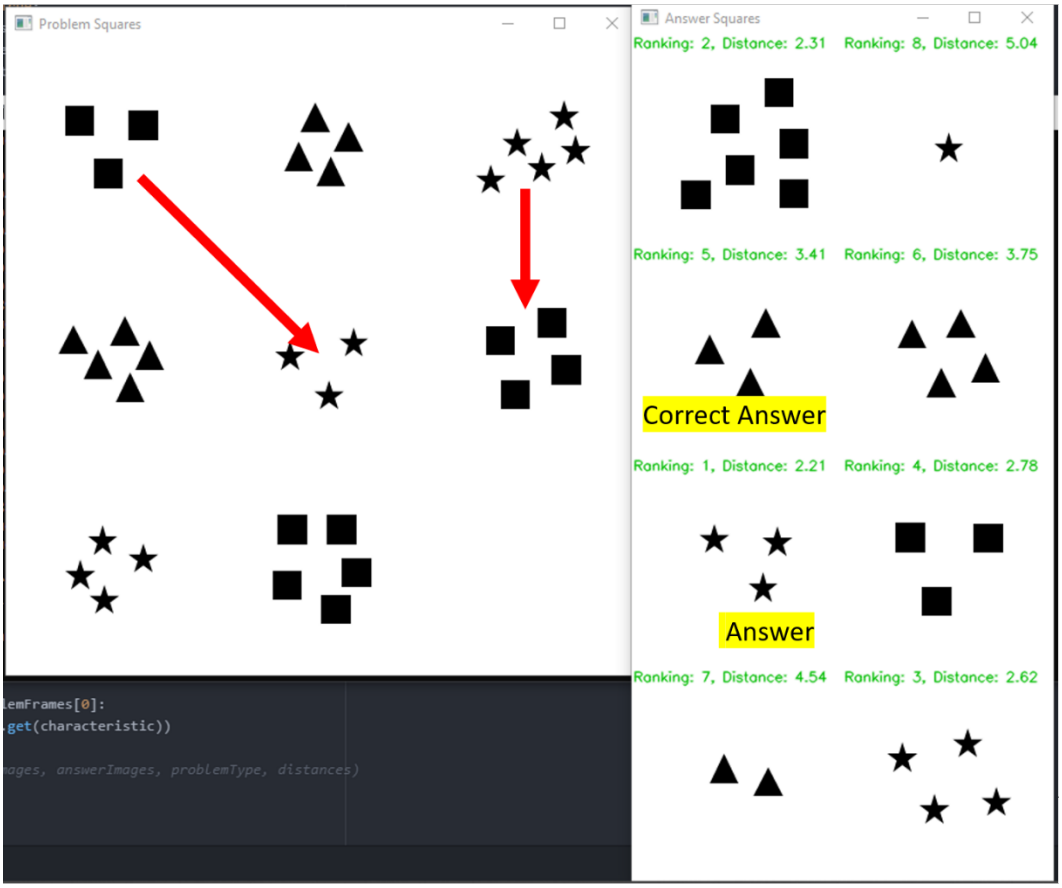

Figure 5- *Set D Weak point*

In this problem, two of the competing transformations do lead to a reduction in the number of shapes or maintain the number of shapes to reach 3 shapes as there are in the correct answer. However, one of the transformations is from square to star, while the other is from star to square. Additionally, the pixel intersection ratio method of distance doesn't apply well for this image because the shapes in the images have substantial differences in alignment. Images we as humans would consider the same are calculated to be largely different. This is a problem the Agent would fare better on with some sense of ontology, or perhaps with a broader solution method involving a row and column level approach.

1.6 Agent functionality- set E

Set E takes a totally different approach from the preceding sets. Set E is the first image plus the 2nd image where overlapping pixels do not add twice, or the first image minus the 2nd image where overlapping pixels are the only to be removed. The Agent does not have a method for the more abstract set E questions but those are the minority.

First, the Agent looks at images B and C to see if there are more pixels in C than B and A. If it is, the Agent concludes this is an addition problem. If not, the Agent concludes subtraction. Then, the Agent adds the black pixels in the H and G frames. If there are any overlapped pixels during addition, the overlapped pixels are once removed to not over add. If there are overlapping pixels during subtraction, the Agent twice removes to subtract only overlapping features. Finally, the Agent selects the answer with the most similar amount of pixels.

1.7 Agent performance

The Agent overall answers 75/96 of the basic and test questions, while answering 23/48 of the Raven's problems. The Agent's best set is set B at 22/24, followed by sets D and C tied at 19/24. The Agent's worst performance is on the underdeveloped set E at 14/24. In general the Agent performs only slightly better the basic questions than the test questions, with the exception of E where it gets 9 basic questions and only 5 test questions.

1.8 Design approach

Most of the design approach was done iteratively from the bottom up. When the Agent failed to solve a problem, I would consider the defining feature change in a problem, and would then add in that feature and test it vs the entire problem set to see how it modified the Agent's overall behavior. If I were to teach the Agent to learn, it would probably be done similarly.

While I often considered how the concepts in the lectures may be used, such as classification, or how more sophisticated image processing techniques may increase the Agent's accuracy, the path of least resistance was often to use more features, change the frames the Agent considers for generating Iexpected, or to change the distance calculations.

There were times when I considered total design overhauls such as considering the 3x3 C and D problems more from a row-by-row and column-by-column perspective, by implementing ontology, or even by using information about the standard deviations of features across a problem to determine when a certain feature was more or less relevant. However, as I already had reasonable performances, I did not pursue those strategies deeply. If I were to create a new Agent, that is the focus I would take.

1.9 Cognitive comparison

The Agent captures some of how I solve these problems as a human. While the Agent understand the image holistically by knowing all of the features that consist of the image, it pays more attention to the differences that happen between the images rather than what the image is. It sees how the differences in other places of the problem create new images, and it takes those differences and applies it itself to create an idea of what an answer would look like. It may not have a perfect idea of the answer, but it can get pretty close. Then, it chooses the answer that is the closest to what it expected to be the answer. This I consider to be a quite human strategy.

What the Agent doesn't do is use ontology, nor does it consider the overall problem holistically. While I answer one of these questions, I think in terms of, "okay so a triangle is rotated at the top row, then in the 2nd row a star is rotated, so then in the 3rd row the Pacman must also rotate". I look at it as a pattern of patterns where I learn the rule to follow deductively, while my Agent is laser focused on a very small set of what is happening within the problem and works as more of a noise filter.

I suspect if my Agent used ontology or more holistic methods, it would also have higher performance, especially as illustrated in Mark Youkey's RPM4 where they received 96/96 on basic and test questions by considering problems from a pattern of pattern perspectives (Youkey, Mark. *Raven's Progressive Matrices Project Milestone 4*). There is much more information baked into the RPM questions that my Agent is not utilizing; it is handicapping itself. My Agent could do more to achieve true *KBAIness*, as well as higher performance, by being more human in its approach.

References

1. Leavitt V.M. (2011). *Raven Progressive Matrices*. In: Kreutzer J.S., DeLuca J., Caplan B. (eds) Encyclopedia of Clinical Neuropsychology. Springer, New York, NY. https://doi.org/10.1007/978-0-387-79948-3_1069
2. Joyner, D.A., Bedwell, D., Graham, C., Lemmon, W., Martínez, Ó., & Goel, A.K. (2015). *Using Human Computation to Acquire Novel Methods for Addressing Visual Analogy Problems on Intelligence Tests*. *ICCC*.
3. Youkey, M. (2021). *Raven's Progressive Matrices Project Milestone* 4. Georgia Tech peer feedback system. https://peerfeedback.gatech.edu/app/feedback/course/210408/assignment/853370/user/13569