Project 1

Diya Benjamin
diyabenjamin@gatech.edu

1 INTRODUCTION:

1.1 Overall idea for addressing the problem:

The 2 x 2 Raven's Progressive Matrices (RPM) problems in Project 1 will be addressed by a knowledge-based AI agent using the **visual approach**. I used Python along with Pillow (PIL) and NumPy libraries for developing the code to build the agent. My agent will use the visual representations of each problem to reason and decide on the correct answer. The general approach will be to break the overall task into number of smaller tasks. Each task consists of a single problem which provides the raw image data as input such as image A, B, C and the 6 answer options. The agent uses these images to analyze and find the answer that matches the most with the transitions between images A, B or A, C.

The analogy in each problem could be solved by comparing two images and finding the transitions that led A to B or A to C. It will then perform the same transition between the third image and each of the 6 option answer images. It will match the option with maximum similarity ratio as the correct answer for that problem. The similarity between two images will be calculated using Tversky's (1977) ratio model of similarity.

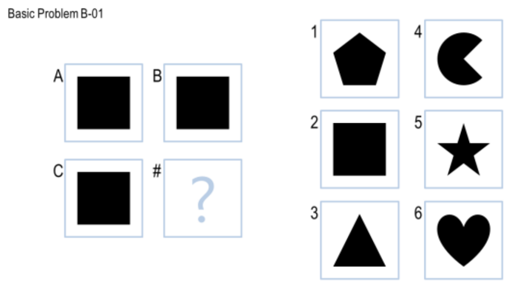

Figure 1. Raven's Progressive Matrices Basic Problem B-01

For testing and debugging, I will initially run the code locally by running RavensProject.py. I will also submit the code to error-check to verify that the code is runnable by auto grader and free of errors. Once the code is able to solve at least 7 out of 12 Basic Problems B, I will do my first submission of the project on Bonnie.

The subsequent submissions will be done whenever significant changes are made to improve the performance of the agent in solving the problems.

2 JOURNAL ENTRIES:

2.1 Submission 1:

Date and Time: 2019-09-14 02:56:25 UTC

This is my first submission and I followed the visual approach to build the agent that can solve the RPM problems. I started with trying to solve the easier problems first such Basic Problem B-01 and B-02. I made use of the Python PIL library to develop the code and used Image and ImageChops classes.

- The agent takes visual images as input and first performs various transformations on the image A to verify if it is matches with image B or C.
- The agent uses the similarity ratio to match the images. It uses the similarity threshold value to decide if the images match. This ratio is derived from the pixel difference ratio of the images using Tversky's (1977) ratio model of similarity which is calculated as shown in Figure 2 below.
- The agent then performs the same transformation on image B or C and verify its similarity ratio with the answer options to find the solution that has the maximum similarity.

$$\boxed{\text{similarity}(A, B) = \frac{f(A \cap B)}{f(A \cup B)}}$$

Figure 2. Tversky's ratio model of similarity equation

The similarity ratio is the complement of difference ratio which is calculated by taking the absolute difference between each pixel from both images and then dividing it with the union of the images.

For Basic B-01 and B-02, the ImageChops module has a difference function which takes two images as input parameters and returns the difference in the images. The getbbox() function on the difference image returns None if the images are identical. I used this function to compare the images A, B and C and return the image from answer options that are identical with these images.

For Reflection, I used the transpose function on an image and input parameter as Image.FLIP_LEFT_RIGHT to mirror flip image A to B. I will then compare the two images using the similarity ratio. If the ratio is greater than threshold value, then apply same transition of flip left-right on image C and match the resultant image to the answer options. The similarity ratio is used to get the most similar match from the options. The threshold for choosing the similarity ratio between options was selected after analyzing their output ratios.

For rotation, I used the rotate function on an image and angle of rotation as the input parameter such as 90, 180 and 270. In order to solve problems similar to Challenge Problem B-02, I used the angles of rotation in increments of 45 such as 45, 90, 135, 180, 225, 270 and 315. The remaining steps are similar to the reflection.

My agent tries to perform similar to humans in solving the problems. It was able to solve problems related to Identity, rotation and revolution. However, it solves the easiest of problems such as Identity quantitatively, which humans could analyze and answer in just a glance. The agent does not recognize the images that are filled and unfilled, which a human would easily analyze and reason to find the correct solution.

For the first submission, I think my agent performed pretty well. The performance of Submission 1 on the types of problems are given in Table 1 below. The execution time was only 36.621 seconds. It was able to provide the correct solution for problems that had identity, rotation and reflection type of problems. It struggled in the problems that had filled and unfilled images such as Basic Problems B-09. The agent also failed in finding the common factors or differences in two images A and B and then applying that difference in image C to get the correct answer solution. These are problems such as Basic B-11 and B-12.

Table 1. Answer solutions for the types of problems for Submission 1.

Submission 1	**Incorrect**	**Skipped**	**Correct**
Basic Problems B	3	0	9
Ravens Problems B	3	2	7
Test Problems B	4	0	8
Challenge Problems B	3	4	5

Submission 2:

Date and Time: 2019-09-17 05:32:31 UTC

When checking for common image components in 2 images (for example in Basic Problem B-12), ImageChops.add for images A and B returns an image and I am checking if that image is either equal to A or B, then perform same operation on images C and all the answer options, and the option with most similarity is chosen as the correct answer. Similarly, I am performing the same steps for ImageChops.multiply operation to solve problems such as Challenge B-09. For Basic Problems B-10 and B-11, inverting the image difference of A and B provides a resultant image which is saved as diff_image. It is then compared with image C by inverting their difference image using ImageChops.difference and ImageChops.invert to get the new_image. Each of the answer options are then compared using similarity ratio with the new_image to check for similarity.

The second version of my agent was performing better than its predecessor. The agent is able to find the differences in images and apply to get the correct answer option. A human would easily be able to find these differences by just looking at the problem images, while the agent needs to break the problem and analyze each image separately. The agent does not recognize the images that are filled and unfilled, which a human would easily be able to analyze and find the solution.

For this submission, I think my agent performed better than the first. The performance of the agent on the types of problems are given in Table 2 below. The execution time was only 36.5021 seconds. It was able to provide the correct solution for problems that had identity, rotation, reflection and deletion of objects. The agent struggled to solve problems that involved shading such as Basic B-09.

Table 2. Answer solutions for the types of problems for Submission 2.

Submission 2	Incorrect	Skipped	Correct
Basic Problems B	1	0	11
Ravens Problems B	0	3	9
Test Problems B	0	1	11
Challenge Problems B	3	5	4

Submission 3:

Date and Time: 2019-09-18 03:36:23 UTC

In order to solve Basic Problem B-09 that involved filled and unfilled image complements, I used the ImageDraw.floodfill() function. It will fill one image with black color and then checks its similarity with second image. If it meets the threshold, then same transformation is applied to third image to find the answer from given options. For rotation, I replaced the lambda function, which was used to get the edges of an image, with ImageChops.invert(). Further, I removed the functions of ImageChops such as multiply() and add() as the difference() function would take care of all those problems.

The agent does not recognize the images that are filled and unfilled, but it tries to solve them quantitatively using ImageDraw.floodfill(). However, a human would easily analyze the same problem by just looking at it to find the correct solution. Overall, I think my agent performed better that the previous submission. The performance of Submission 3 on the types of problems are given in Table 3 below. The execution time was reduced to 33.0413 seconds since I combined several functions into a single function. It was able to provide the correct solution for problems that had identity, rotation, reflection, filled shapes and deletion of objects.

Table 3. Answer solutions for the types of problems for Submission 3.

Submission 3	Incorrect	Skipped	Correct
Basic Problems B	0	1	11
Ravens Problems B	0	3	9
Test Problems B	0	3	9
Challenge Problems B	0	10	2

Submission 4:

Date and Time: 2019-09-18 03:43:02 UTC

When using invert() in the rotation function, the results regressed as it did not satisfy the expected requirement of getting edges. The agent was was either skipping the rotation related problem or providing incorrect output. So, I reverted this back to lambda which made a lot of difference to the output.

Though the agent does not perform as efficiently as a human, it was able to solve all the 12 Basic Problems. The performance of Submission 4 on the types of problems are given in Table 4 below. The execution time was only 36.5021 seconds. Also, from the 4 submissions, we can see that the runtime remains almost the same, but never increases drastically. This shows the efficiency of the agent in always taking the same amount of time regardless of how many problems were solved. Thus, we can represent the agent's Big O notation as almost equal to O(1).

Table 4. Answer solutions for the types of problems for Submission 4.

Submission 4	Incorrect	Skipped	Correct
Basic Problems B	0	0	12
Ravens Problems B	0	3	9
Test Problems B	0	2	10
Challenge Problems B	1	7	4

Submission 5:

Date and Time: 2019-09-20 14:48:34 UTC

I gave more priority to the function that finds the difference ImageChops.difference() between images to get the correct answer option. Along with problems B-10 and B-11, it also solved the identity problems such as B-01, B-02 and B-03. The priority for each function is identified through their arrangements and ordering. A problem passes through the first function in line which has the highest priority. If it does not return anything, it is passed to the next function in which has comparatively lesser priority than the first.

To solve Challenge Problem B-08, where number of vertices of the object increases from A to B, I used a function to count the number of vertices in each image. I took the difference between the counts to add it to number of vertices in C. The resultant count was compared with the vertices count of each answer option.

For the 5th submission, I believe my agent performed pretty well and fast too. The performance of Submission 5 on the types of problems are given in Table 5 below. The execution time was reduced to 30.6477 seconds. It was able to provide the correct solution for problems on identity, rotation, reflection, fill objects, deletion and shape change.

Table 5. Answer solutions for the types of problems for Submission 5.

Submission 5	Incorrect	Skipped	Correct
Basic Problems B	0	0	12
Ravens Problems B	1	2	9
Test Problems B	0	3	9
Challenge Problems B	1	5	6

Submission 6:

Date and Time: 2019-09-22 18:49:12 UTC

In this submission, I have concentrated more on optimizing the code and adding each type of solution check in separate functions. I setup global variables for the input images. A variable named 'option' is initialized to -1 in the Solve() method. Whenever a function is called, its return is saved in 'option' variable. As the solve_image_difference() method has highest weightage, it is called first. When the problem passes through this method, it returns an option and skips all remaining functions.

If the problem does not pass through any function, the option variable does not change and hence returns -1. Each function call will check for these conditions. I have defined two constants for threshold such LOW_SIMILARITY_THRESHOLD and HIGH_SIMILARITY_THRESHOLD and initialized them with values 0.97 and 0.98 respectively.

My agent performs much better than the previous submissions. It however does not think the way humans do. The agent takes a different approach where it breaks the problem into multiple steps. The agent fails to solve few problems in the Challenge type that has shaded images.

For this submission, I believe my agent performed well as the code was highly optimized compared to previous submissions. The performance of Submission 6 on the types of problems are given in Table 6 below. The execution time was only 33.6289 seconds. It was able to provide the correct solutions for 12/12 Basic Problems and 10/12 Test Problems.

Table 6. Answer solutions for the types of problems for Submission 6.

Submission 6	Incorrect	Skipped	Correct
Basic Problems B	0	0	12
Ravens Problems B	1	2	9
Test Problems B	0	2	10
Challenge Problems B	1	5	6

3 CONCLUSION:

For designing my agent, I followed the process of targeting one type of problem at a time. The initial Basic problems such as B-01, B-02 and B-03 were easy to analyze which provided a basic foundation to building the agent. From there I picked up remaining problems one by one, understanding its type and reasoning with the solution. I started to think how humans would solve these problems step by step and followed the same approach in developing the agent. This process displayed deliberate improvement between consecutive submissions.

My final agent is somewhat similar to how a human would solve these problems. However, the agent requires some more improvement in order to reach the level of human cognition. The agent solves the problem in the say way the humans do. They break each problem down to simple steps to decide on the correct answer. They identify the matching objects in the image and decide the ways in which one image projects onto another. Agent solves the problems by evaluating each transformation quantitatively, which in some ways makes it better efficient than humans.

The mappings between the images and their transformations can be compared to a semantic network. These mappings are identified through Generate and Test method by developing a complete set of all possible mappings and then comparing them using similarity ratio to identify the best one. The smart generator learns and generates transformed image from best similar transformation and Tester tests compare the transformation of image on the available answer options using the threshold limit to decide on the answer.

The agent does not have the ability to learn and identify new transformations from the existing ones, even though it has all the information available. The agent sometimes cannot perform similar to humans in solving these problems. It tries to find the correct answer based on only the translations and pixel similarity ratio. If there is only a minor change in the sections of images in answer options, it leads to ambiguity resulting in selection of inaccurate answer by the agent. Similarly, humans can cut-short their problem evaluation, if they feel the transformations in each image cannot help evaluate the correct solution.

If given more time and resources, I would improve my agent to look at multiple transformations at the same time, just like humans. I would solve the problems that involve removing of alternate circles for transitioning from image A to B, and also the shaded objects which are covered in Challenge Problems. I would also improve on the metric used to check the similarity between images to improve the agent's accuracy and performance. I would also try to improve the mapping between objects especially in problems that have multiple objects in an image.

REFERENCES

1. Goel, Ashok. Joyner, David. Thaker, Bhavin. KBAI eBook: Knowledge-based Artificial Intelligence October 6,2016 [Retrieved From] https://files.t-square.gatech.edu/access/content/group/gtc-bf70-dcdf-5e4b-a676-3e2fb8df4997/KBAI-Ebook/kbai_ebook.pdf
2. Kunda, Maithilee. McGreggor, Keith. Goel, Ashok. June 2013. A computational model for solving problems from the Raven's Progressive Matrices intelligence test using iconic visual representations. https://www.sciencedirect.com/science/article/pii/S1389041712000423