Solving Raven's Progressive Matrices: An AI Approach

Simon Benzer
simonhbenzer@gmail.com

Abstract—This paper outlines and discusses the methodologies, performance, and efficiency of an agent that is able to correctly solve 65 of 127 Raven's Progressive Matrices.

1 METHODOLOGIES

My agent functions by performing 5 main methods, the latter 4 focused on solving particular types of problems, and the former on preparing images for analysis. Those methods are geared at solving rotation patterns, bitwise patterns, flip patterns, and a voting method utilizing Intersection Pixel Ratio (IPR), Dark Pixel Ratio (IPR), Contour Count, and Contour Density metrics for solving the remaining problems. The preparation method is for removing noise and ensuring each image is properly prepared for analysis.

1.1 Preprocessing/Noise Reduction

Once it was determined that actual problems contained noise in their images, a noise-reduction methodology was critical. Noise reduction is performed in 3 steps. First the image is forcefully transformed into greyscale to ensure no RGB-colored pixels are present. Then, images are thresholded—in other words, all pixels below a specific grey-threshold (namely 100), are cast to white pixels (threshold of 0), and those above are cast to black pixels (threshold of 255). Then the image is blurred to reduce noise using OpenCV's Gaussian Blur function. See Appendix 8.1: Preprocessing/Noise Reduction for an example.

1.2 Rotation Patterns

In order to solve matrices with rotation pattern solutions, my agent rotates the images in the problem set, and checks if the final image in the row/column is a rotation of an adjacent cell. If so, it rotates the adjacent images of the missing answer—Cell I in 3x3 matrices, and Cell D in 2x2—thus determining what the answer should look like. It will then compare the generated answer to all answers

in the provided answers, if one matches the generated answer within the threshold, it is selected.

In order to do this, the agent utilizes mean squared error (MSE) between images through the NumPy library, and if the MSE is less than the chosen precision it selects that answer. The MSE precision chosen was 0.05 based off extensive local testing on what solved the most answers correctly.

1.3 Bitwise Operation (and, or, xor, not) Patterns

If a rotation pattern is not found, and the matrix is a 3x3, the agent then checks if the pattern is a bitwise operation. To do that, it performs each bitwise operation—and, or, xor, and not—on each pair of adjacent cells, and checks if the third cell matches what it generated using MSE—like in 1.1, the precision is similarly 0.05. If the generated pattern matches the 3rd cell in the row/column, the pattern used is determined as the pattern, and then it performs that operation on either the last row or last column, based on whether the pattern was found in a column or row. If an answer matches the generated pattern under the precision threshold, it selects the answer, if not, the agent moves on. Like in section 1.2, both the MSE and MSE precision threshold are performed the same way.

1.4 Flip Patterns

If both a rotation pattern *and* a bitwise pattern are not found, the agent checks for a flip pattern. The agent first generates a flipped image of a cell, and then checks if an adjacent cell matches that generated image. If so, the pattern is determined to be flipping, and an answer is generated for the missing cell based on whether the flip pattern was found in a column or a row. The agent checks for flipping of images along their x-axis, along their y-axis, and along both their x and y-axes simultaneously.

Like in sections 1.2 and 1.3, MSE, and its precision threshold of 0.05, is utilized to determine whether a generated image is similar enough to a possible answer/adjacent cell for us to conclude the pattern is correct.

1.5 IPR, DPR, and Contour Weighted Voting Method

If the methodologies outlined in sections 1.2-1.4 fail to generate an answer with enough precision, the agent shifts to its main methodology: a voting heuristic.

To do this, the agent first generates 4 metrics (See Figure 1) for each pair of adjacent cells,, then it generates these 4 metrics for each answer choice and the cells adjacent to that answer choice. For 2x2 matrices, this is one pair (D to C), and for 3x3 matrices this is 3 (I to F, I to H, and I to E).

Using a threshold of 5% difference, if an answer pair's metric is within the threshold to another pair's metric, a vote is cast for that answer as the correct answer. The answer with the most votes is chosen as the correct answer.

Metric	**Definition**
Intersection Pixel Ratio (IPR)	The percentage of dark pixels that share the same location in both cells versus those that don't.
Dark Pixel Ratio (DPR)	The percentage of cells that are dark in both cells versus those that aren't
Contour Density Delta	The difference in total area percentage (pixels in shapes over all pixels in cell) of all shapes found in one cell versus the other.
Contour Count Delta	The difference in number of shapes between cells.

Figure 1—Metrics used in weighted voting method; IPR, DPR, Contour Density Delta, and Contour Count Delta.

2 AGENT PERFORMNCE

My agent performs decently well on all sets of problems except D, and all subsets except Challenge (Figure 2).

SET	**BASIC**	**TEST**	**CHALLENGE**	**RAVEN'S**
B	8/12	7/12	2/12	6/12
C	5/12	4/12	1/12	5/12
D	3/12	4/12	1/12	2/12

E	8/12	6/12	1/12	2/12

Figure 2— Agent performance on each problem set, and subsequent subset, in the final Raven's Project challenge.

The total time runtime for the agent was 4.283556709289551 seconds, with an average of 0.022310193216 seconds per problem; a more than satisfactory result.

3 PROBLEMS SOLVES SUCCESSFULLY

Problems my agent solves correctly can accurately be described by 4 main descriptors: rotation-based patterns, bitwise operation-based patterns, flip-based patterns, and problems with patterns easily described by the metrics described in Figure 1—in other words, a "voting" problem. The following figures, 3-5, illustrate 3 problems my agent can correctly solve, using the aforementioned descriptors.

In Figure 3, one will notice that there is, in effect, *no pattern*. Nothing changes between cells A and B. However, the agent correctly solves this problem using its rotation-based approach. This is because the agent determines that this is a rotation of 360 degrees.

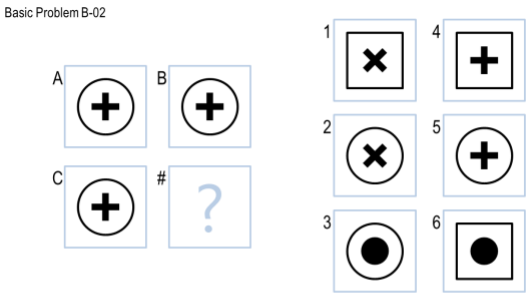

Figure 3—Basic Problem B-02; a "rotation" problem. The agent determines the problem to be a rotation of 360 degrees.

In Figure 4, one will notice that there is, in effect, *no pattern*. Nothing changes between cells A and B. However, the agent correctly solves this problem using its rotation-based approach. This is because the agent determines that this is a rotation of 360 degrees.

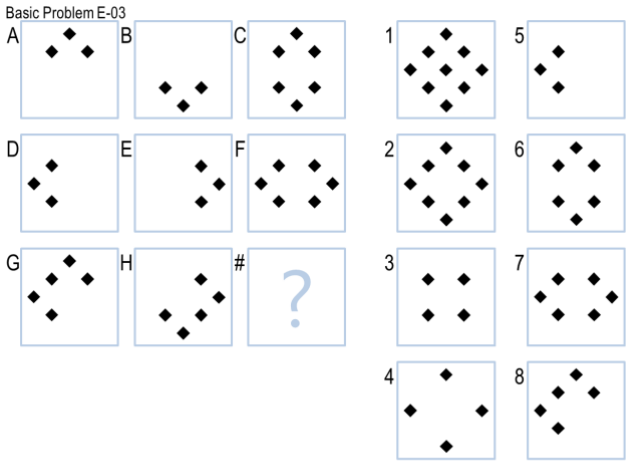

Figure 4— Basic Problem E-02; a "Bitwise" problem. The agent solves this problem by utilizing the bitwise "OR" operator.

Lastly, one will notice that Figure 5 isn't solvable using flip patterns, rotation patterns, or bitwise operations. As a result, the agent must shift to its voting methodology. The agent calculates the 4 metrics outlined in Figure 1, and casts a vote for each answer that's adjacency pair's metric falls within the 5% threshold of another pair. The answer with the most votes, in this case answer 1, is chosen.

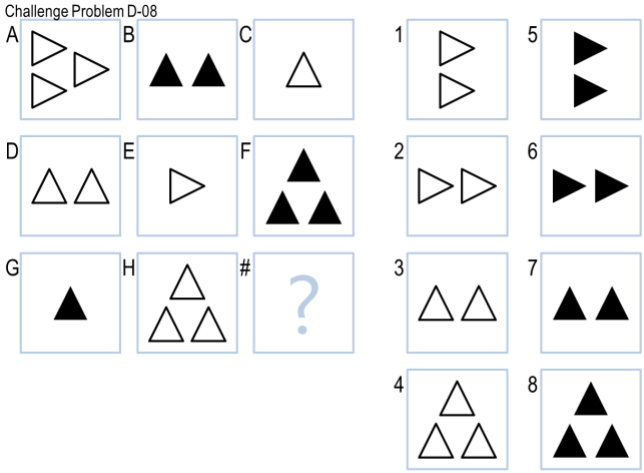

Figure 5— Challenge Problem D-08; a "voting" problem. The agent is unable to solve the problem using flip, rotation, or bitwise operations, and instead solves it using its voting methodology.

4 PROBLEMS SOLVES UNSUCCESSFULLY

My agent struggles in a number of ways, however the most significant are ways that particularly fall into the flaws of its current IPR/DPR/Contour methodology. If a problem is not a rotation, bitwise, or flip pattern, it relies on the voting methodology to solve it. Each methodology has flaws however, and those flaws are explored further below (Figures 6-7) in depth.

First, we can look at Challenge Problem D-10 (Figure 6); the solution to Figure 6 is 5, however the agent believes it to be 2. The agent first attempts to solve it using rotation, but fails. This is because, while the solution *is* actually a rotation solution, the agent looks at cells A to D, and notices a 180 degree rotation, however cell G is *not* a 180 degree rotation of G, and is instead actually a rotation of 270 degrees. Unfortunately, the agent is able to see that answer 2 is a 180 degree rotation of F with a MSE of less than 0.05, and chooses that as its guess.

This is a clear example of a flaw in the logic on the agent. The agent is incapable of recognizing rotation patterns that are not the same across all adjacent cells in a row/column, instead of the addition of an extra 90 degrees to the rotation as is shown here.

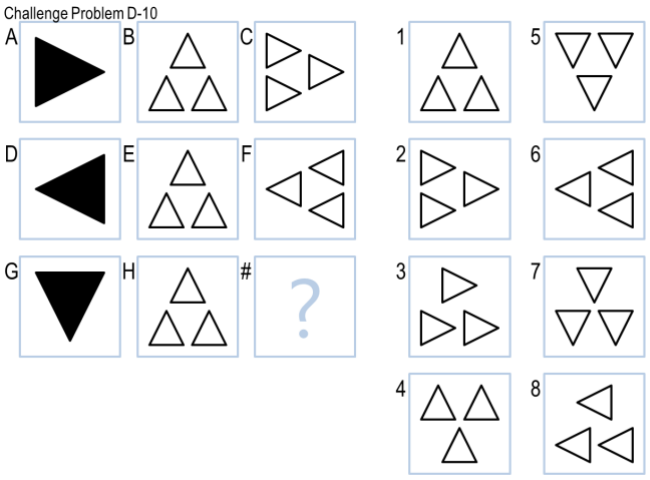

Figure 6— Challenge Problem D-10; a "rotation" problem. The agent incorrectly believes that the answer is 2 because it is a 180 degree rotation of F, like F is to C. The correct answer is 5.

Challenge problem E-04 is a great example of multiple failure points. In this instance, the solution is 5, however the agent determines it to be 4 using its voting methodology.

First, the agent should have caught it as a bitwise OR operation, however it did not. This is because the agent checks for bitwise operations in 2 ways: an operation—and, or, xor, not—of A and D becoming G, and an operation of A and B becoming C. This could have been resolved by also checking in the opposite directions—e.g. C and B becoming A. It did not find an answer however, and thus moved on to the voting methodology.

Second, the agent should have determined the correct answer using its voting methodology, instead it determined the answer to be 4. The agent looked at the DPRs, IPRs, Contour Density Deltas, and Counter Count Deltas (Figure 1), and determined that the answer with the most votes was answer 4. The agent sees 4's relationship with C and A, as well as a similarity with D, and determines it to be the answer with the most likelihood of success. While not inherently a flaw in logic, this indicates that the agent's voting methodology needs to either account for more metrics, or that votes might need to be weighted differently.

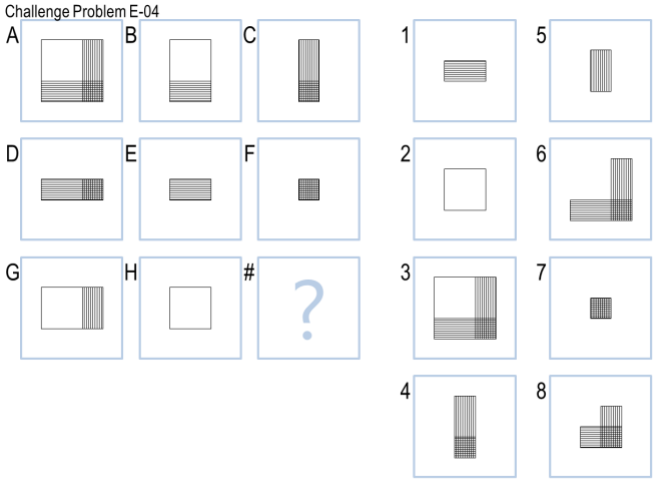

Figure 7— Challenge Problem E-04; a "bitwise" problem. The agent incorrectly believes that the answer is 4, but the answer is 5.

5 DISCUSSION

I would describe my overall approach to designing my agent as a combination of two distinct methodologies: designing a core method for solving all problems that can be continuously built upon and enhancing that method with supplementary methods for solving distinct patterns. The purpose of this was two-fold: one, to be able to have a reliable approach to solve any and all matrices, and two, to enhance that approach by being able to quickly recognize and solve specific types of problems it knows it will encounter. This way the agent has methods to enhance its efficiency, while at the same time having a reliable and effective method if/when those heuristics fail.

This is based on what I desired to be a combination of the pros and cons of human behavior; humans are very good at recognizing patterns they know, and even many they don't, but when we can't find a noticeable pattern, we have a tendency to flounder. This agent is supposed to take the good—our ability to recognize and act based on known patterns—and fix the bad—our tendency to flounder when we can't—by having a reliable method for when *it* can't.

When I first built my agent in Milestone 2, I focused on designing an IPR/DPR method similar to the one described in Joyner, 2018. This method calculated the IPRs and DPRs of a few vertically and horizontally adjacent cells—B to C and D to G—and checked if the Euclidean distance of answer pairs was within a range of 0.1 to problem pairs, if so, it cast a vote in favor of that answer choice. This became the foundation for my approach. Subsequent milestones focused on two things: enhancing the IPR/DPR approach and adding heuristics.

Each milestone added new heuristics to check for: rotation patterns, then bitwise patterns, and finally flip patterns. Each milestone also enhanced the IPR/DPR method. First it added more adjacency pairs—A to B, A to D, A to E, B to E, D to E, E to F, G to H, and E to—then it simplified the voting method for 3x3 matrices from a complex 4-dimensional Euclidean distance comparison to an increased number of 2-dimensional comparisons. In the Final Project, I enhanced the voting method even further by adding 2 more metrics—Contour Density Delta and

Contour Count Delta—and scrapped the Euclidean distance calculations entirely. Instead, each metric was similarly calculated for each adjacency pair, and each adjacency pair for the answer choices, but each answer only received a vote if their metric was within 5% of the metric of another pair's metric.

Ultimately, my agent became what I desired: one—arguably—reliable methodology for solving any problem supplemented by specific heuristic based approaches to specific problems.

6 IS THIS AI AGENT HUMAN-LIKE?

Because my agent is based on a two-fold approach—a general methodology supported by heuristic checks—it is both human-like and not in its approach to Raven's Progressive Matrices. I use myself as an example of a human in this section, thus in any mention of what a human does or might do, the foundation for said belief is how I react and respond to that scenario.

First, I would argue that it's heuristic-based approaches—rotation, flip, and bitwise—are quite humanlike. Human's look for patterns, and once they learn to recognize a pattern, they are more likely to recognize it in the future; in fact the brain even subconsciously looks for it. In this way, my agent learned, like a human, to recognize patterns and similarities between cells.

Like a human, the agent is capable of performing those patterns on an object, analyzing what those results would look like, and then determining if that result is similar enough to other cells, or other answer choices, that it believes it to be a likely answer. For example, if a human was to look at Figure 8, they would first look at the relationship between A and B, and likely recognize that B is flipped along the y-axis. The agent would do the same. A human would then look at C and would attempt to do the same flip along the y-axis, resulting in answer 1. The agent did the same. In this regard, I think my agent is very human-like.

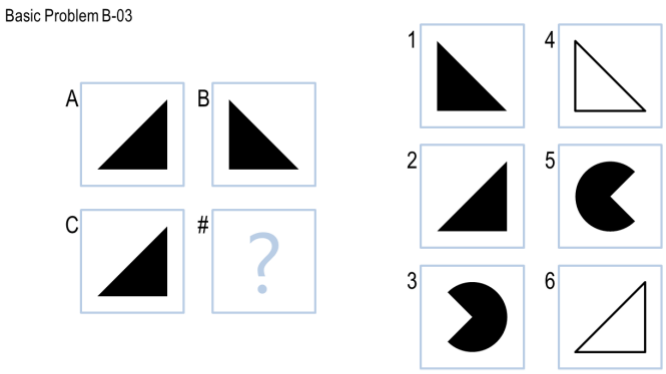

Figure 8— Basic Problem B-oe; a "rotation" problem. The agent correctly determines the answer to be 1.

My agent's main approach, the IPR/DPR/Contour voting method, is in my opinion not very humanlike. Humans rarely calculate, as my agent does, distinct metrics of pixel count, number of shapes, the density of those shapes, etc. and then internally vote on how many times an answer choice is within range of those metrics, thus attempting to generate an unbiased answer.

A human may, in many cases, attempt to determine an answer based on those choices individually, like filling a shape or adding a shape, or even together, adding a shape and then filling it. However, what humans do not do, is take all those metrics—change in dark pixels, change in intersecting pixels, change in number of shapes, change in areas of those shapes—and in an unbiased way vote on these metrics. This is for many reasons, most significant of them being the time it would require doing by hand, and human's tendency to give certain metrics/relationships weight over others. Humans are innately biased creatures; my agent is not.

7 REFERENCES

1. Joyner, D.A., Bedwell, D., Graham, C., Lemmon, W., Martinez O., & Goel, A.K. (2015). Using Human Computation to Acquire Novel Methods for Addressing Visual Analogy Problems on Intelligence Tests. *International Conference on Innovative Computing and Cloud Computing.*

8 APPENDICES

8.1 Preprocessing/Noise Reduction

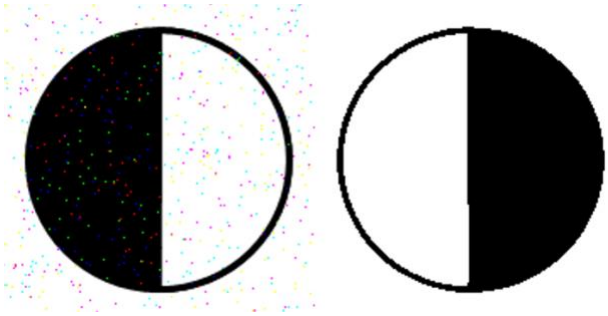

Figure 9— Image before and after noise reduction has been performed. Observed noise was both artificially generated and colored.