Report of Project 1

ZHENG FU
zfu66@gatech.edu

Abstract. In this report, I describe the algorithms underlying a knowledge-based AI agent for solving basic 2 x 2 puzzles in Raven's Progressive Matrices test. A further discussion was made to elucidate the difference between the AI agent behaviors and the ways a human using to solve these problems.

Section I: Provide a narrative of how you approached the project.

In general, there are two ways to solve the basic 2 x 2 puzzles in Raven's Progressive Matrices test. One is to estimate the difference between figure A and B, delta(A,B), then compare figure C with each candidate figure d and estimate their differences delta(C,d), followed by measuring the similarity between delta(A,B) and delta(C,d) and made a decision. This methodology is called *infer-infer-compare*. The other approach is also gauging the delta(A,B) first, but then applied the delta(A,B) to figure C and infer the most possible answer of figure D. This methodology is called *infer-map-apply*[1].

In this project my approach to figure out the basic 2 x 2 puzzles was like a mixture of *infer-infer-compare* and *infer-map-apply*. First of all, the AI agent defines the rules of transforming figure A to B, **rules(A,B)**, and the rules of transforming figure A to C, **rules(A,C)**. Secondly, for each candidate figure d, the AI agent defines the rules of transforming figure B to d, **rules(B,d)**, and the rules of transforming figure C to d, **rules(C,d)**. Finally the AI agent estimates the similarity between **rules(A,B)** and **rules(C,d)**, as well as the similarity of **rules(A,C)** and **rules(B,d)** and generates a score. The candidate figure with the maximum score will be selected as the final solution.

How did your approach change over time? What modifications did you make?

In the first round of designing AI agent, I calculated the difference between the number of objects in figure A and that of figure B, **objects_num_diff(A,B)**. Then I calculated the difference between the number of objects in figure C and that of each candidate figure d, **objects_num_diff(C,d)**. If **objects_num_diff(A,B)** is not equal to **objects_num_diff(C,d)**, such candidate(s) will be removed[2]. After filtering, if more than one candidate exist, the AI agent will select the maximum figure number from the candidates' pool as the final answer. This approach is quite useful for certain puzzles. From Figure 1 we could see the objects number difference between figure A and B in puzzle B-12 is 2 (5 rings – 3 rings), and the only correct answer is figure 1 since figur

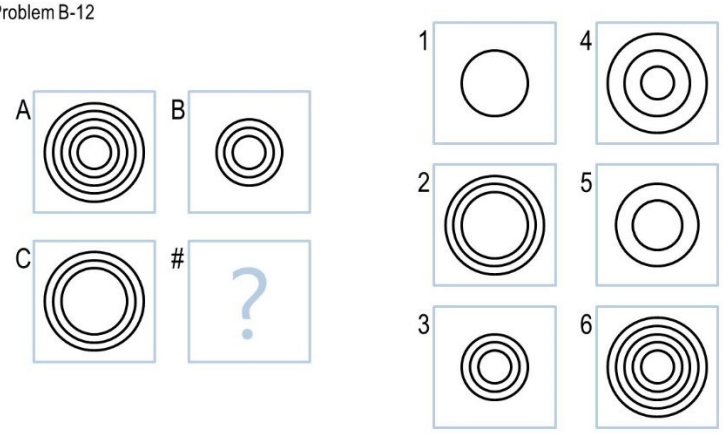

Figure 1. Basic Problem B-12

The second round of my implementations is to consider solving the simplest problem: figure A, B and C only have one object, like Basic Problem B-01 and B-03. The third round and fourth round of improvements focus on adding / removing one object to / from figure A while figure A only has two objects. The final round of modifications is to solve problems that figure A, B and C have two objects, like Basic Problem B-02, and the AI agent performs well.

Section II: Specifically for your final agent, how does it work?

What is its process of solving the problem? How does it represent information about the problem?

As far as my final agent was concerned, its process of solving the problem could be divided into two major parts: 1) Determining the rules of transformation between two figures and 2) Scoring the similarity between rules.

1. Rules of transformation.

The rules of transformation could be categorized into few groups according to the object's attributes.

1). Attribute group 1: 'above' and 'overlap'.

The rules of these attributes are simply binary, just check if they exist or not.

2). Attribute group 2: 'size', 'shape', 'inside' and 'fill'.

The rules on here are testing if these attributes were changed during the transformation. Note that the values of size are categorical variables and the AI agent converts them into continuous variables. Hense the size difference between two objects could be represented by an integer.

3). Attribute group 3: 'angle'.

The rules for 'angle' focus on objects rotations. If the sum of object's angle in figure A and that in figure C = 360 degree, and the sum of object's angle in figure B and that in figure D = 360 degree, the objects are symmetry with respect to x-axis. Similarly, if the sum of object's angle in figure A and that in figure B = 180 or 540 degree, and the sum of object's angle in figure C and that in figure D = 180 or 540 degree, the objects are symmetry with respect to y-axis.

4). Attribute group 4: 'alignment'.

The rules for 'alignment' are complicated since the AI agent must consider both rotations and shiftings. Here I designed a numbering system for alignment as shown in Figure 2. Top-left = 0, Top-right = 1, Bottom-left = 2 and Bottom-right =3, respectively.

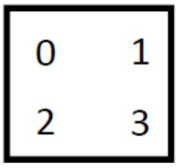

Figure 2. Numbering system of alignment.

Therefore if **alignment_diff(A,B)** is equal to **alignment_diff(C,D)** and **alignment_diff(A,C)** is equal to **alignment_diff(B,D)**, the AI agent will believe the objects were in alignment. For instance, in Basic Problem B-05 both figure 4 and 5 are possible answers. Because **alignment_diff(A,B) =** 3(Bottom-right) – 2(Bottom-left) = 1 is equal to **alignment_diff(C,D) =** 1(Top-right) – 0(Top-left) = 1, and **alignment_diff(A,C) =** 3(Bottom-right) – 1(Top-right) = 2 is equal to **alignment_diff(B,D) =** 2(Bottom-left) – 0(Top-left) = 2 (see Figure 3).

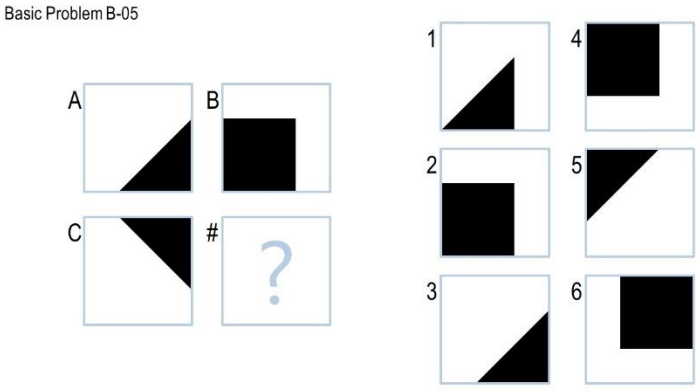

Figure 3. Basic Problem B-05

2. Scoring similarity.

All transformation rules were saved in hash-tables using attributes as keys and the initial total score was setup to zero. For attributes in group 1 and 2 ('above', 'overlap', 'fill', 'shape', 'size' and 'inside'), the AI agent first compares the transformation from figure A to B with the transformation from figure C to D. For a given hash key (the attribute), if the corresponding values of two hash tables are identical, the total score will add 1. Then the AI agent will do the same thing for transformation from figure A to C and transformation from figure B to D.

For attribute 'angle' and 'alignment', the AI agent needs to consider transformation figure A to figure B, figure C to figure D, figure A to figure C and figure B to figure D at the same time, to ensure the symmetry and alignment with respect to both x-axis and y-axis. One example is Basic Problem B-04. If the AI agent believes figure A clockwise rotates 90 degree and becomes figure B, then the correct answer should be figure 1 (figure C clockwise rotates 90 degree). If the AI believes figure A anti-clockwise rotates 90 degree and becomes figure C, then the correct answer is still figure 1. Only after the AI agent considers all horizontal and vertical transformations, it will choose figure 3, which is the true key and closer to human cognition since we would like to see a white square shape in the middle if we select figure 3.

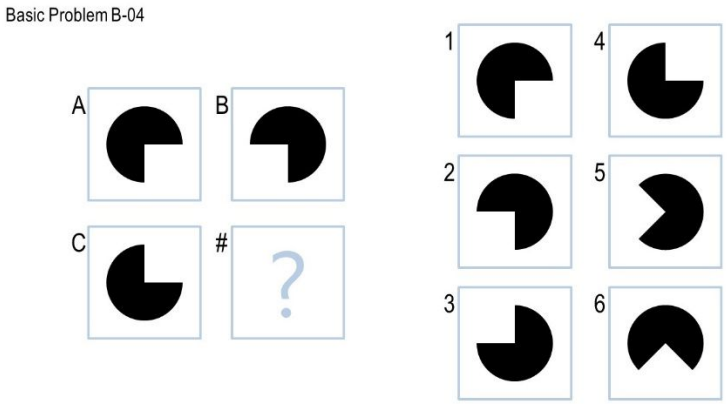

Figure 4. Basic Problem B-04.

Section III: Specifically for your final agent, describe its performance.

How many problems does it answer correctly?

The following table summarizes the accuracy of the final AI agent.

Table 1. The accuracy of the final AI agent for Problem Set B.

Problem Type	Correct	Skipped	Incorrect
Basic	12	0	0
Test	9	0	3
Challenge	0	12	0
Ravens	0	12	0

How efficient is it?

The AI agent uses 6.01768 seconds to solve the Problem Set B, so I believe it is quite efficient.

How general is it?

The AI agent performs well in Basic and Test problems yet skips all Challenge and Ravens questions.

Does its performance on the Basic and Test sets differ significantly, or are they about the same?

The AI agent's performance is moderately different on the Basic and Test sets. On Basic set it has 100% accuracy whereas on the Test sets it has 75% accuracy (not bad actually).

Section IV: Specifically for your final agent, what are its limitations?

What types of problems does it currently answer incorrectly?

The final AI agent answers all questions in Basic Set correctly. And due to the lack of information about Test Set, I do not know what types of problems in Test Set it did not answer correctly.

If it currently answers all or almost all problems correctly, what kinds of problems would it struggle with?

According to the design of my final AI agent, there are three kinds of problems it will struggle with:

1. Figure C has more objects than figure A, like Basic Problem B-11. In this scenario the agent will select the maximum figure number from the candidates' pool as the final answer.
2. Figure A has at least two objects and add/remove one object to/from figure A to transform to either figure B or C. The agent will skip such kinds of questions.
3. Adding/removing at least two objects to/from figure A to transform to either figure B or C. The agent will skip such kinds of questions.

Section V: For both your final agent and your entire design process, connect your project to human cognition.

I neither feel that the nature of my revisions reflecting the way a human learns from experiences, nor feel my final agent solves the problems similar to how a human would do so. The reason is AI agent does not have human eyes, thus it could not really 'see' the figures with human vision. The inputs of the AI agent are structured / unstructured text data and how to interpret them determines its performance. For example, I did not implement any function to map the objects within a figure, like using 'above' and 'overlap' attributes. Although they are very important and useful, it is quite difficult to make AI agent fully understanding such attributes and applying them to infer the solution. However, as human beings we could identify one object is 'above'/ 'overlap' another object at a first glance, and quickly employ such information to search for correct answer.

Another example is shown in Figure 5. According to my alignment numbering system mentioned in Section II, figure D may not be a correct answer. Because **alignment_diff(A,B) =** 3(Bottom-right) – 2(Bottom-left) != **alignment_diff(C,D) =** 2(Bottom-left) – 3(Bottom-right), and **alignment_diff(A,C) =** 3(Bottom-right) – 2(Bottom-left) is not equal to **alignment_diff(B,D) =** 2(Bottom-left) – 3(Bottom-right) either. However, a few people including me may think figure D is a solution since the alignment of objects in figure A and B are zoom-in and transform to the alignment of objects in figure C and D. The fact is our human eyes are like camera and capture objects zoom-in/zoom-out easily, yet it is difficult to train the AI agent to understand and apply such concepts. In summary, it is still a long way to build and train AI agent who could really mimic human cognition process in solving Raven's Progressive Matrices problems.

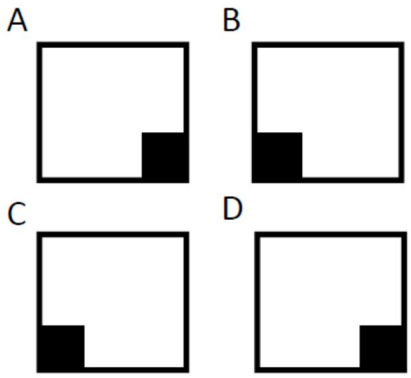

Figure 5. An example of objects' alignment zoom-in.

References

1. Lovett, A., & Forbus, K. (2012). Modeling multiple strategies for solving geometric analogy problems. In *34th Annual Conference of the Cognitive Science Society*. Sapporo, Japan.
2. Evans, T. G. (1964). A heuristic program to solve geometric-analogy problems. In *AFIPS '64 (Spring) Proceedings of the April 21-23, 1964, spring joint computer conference.* Washington, D.C., USA.