Jonathan Louie
CS 7637

Project 2 Reflection

How does your agent reason over the problems it receives? What is its overall problem-solving process? Did you take any risks in the design of your agent, and did those risks pay off?

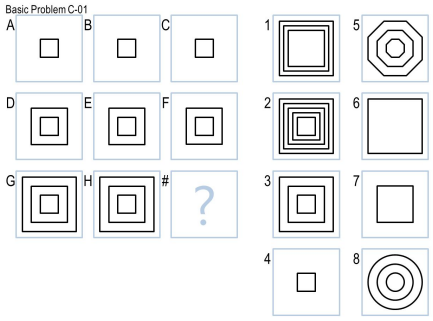
My agent solves 3x3 Ravens Progressive Matrices (RPMs) by searching its bank of pre-programmed patterns and seeing if any such pattern fits the current problem it's trying to solve. This process was modeled after my own personal way of thinking about RPMs.

After looking through the set of RPMs given to us for this project, I found that there were a few different "levels" of patterns and problem-solving skills used. For example, for some problems, the solution was the figure that matched the transformation patterns found across the rows and down the columns of the matrix. Other times, the solution was the figure that completed the larger image, i.e. granted the matrix symmetry of some sort. When designing my agent, I wanted to take this varying depth of reasoning into consideration and implement it into my agent.

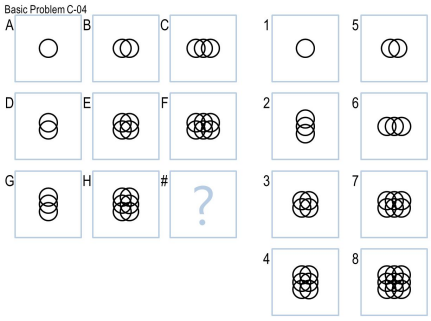
In this pursuit, I risked using visual data representation, as opposed to the verbal representation, as I believed it would give me the greatest control and allow me to replicate the most human-like behavior. Though it did grant me the flexibility I wanted, it undoubtedly more difficult to manipulate which led to an overall loss in performance, mostly due to the increased cost of development.

Using this visual representation, my agent was able to choose the most appropriate of the following patterns and used the selected pattern to find a solution:

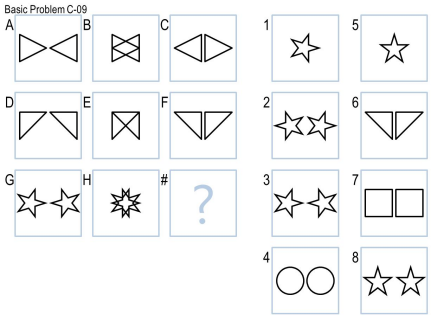
- Vertical or horizontal symmetry across the entire matrix, i.e. treating the matrix as one large image (Problem C-01, Problem C-07)
- Multiplying and translating the image as you move left to right in the matrix of figures (Problem C-04)
- Objects scaling in place (Problem C-06)
- Objects translating horizontally past each other (Problem C-09)



Problem C-01



Problem C-04



Problem C-09

My agent always tried to fit RPM at hand with most holistic patterns first, just as I would. These "bigger picture" patterns are always the most obvious to humans so it should be the same with my agent as well. As the patterns get more specific in behavior, they were pushed further down the pattern-finding search queue. If the problem did not fit any of the pre-programmed patterns, no solution was given.

How does your agent actually select an answer to a given problem? What metrics, if any, does it use to evaluate potential answers? Does it select only the exact correct answer, or does it rate answers on a more continuous scale?

My agent has a list of pre-programmed patterns that it uses. The list is ordered from most holistic to least holistic. The first pattern that the agent believes fits the problem is used to pick a solution. Once a pattern was found, this pattern was used to generate a figure that was expected to match one of the proposed solutions. All potential solutions were checked against the predicted solution to see if any of them matched. The solution most closely matching the predicted solution was chosen.

For a solution to be considered a "match", it had to have a pixel match of 98% or higher. In cases where more than one solution met this criterion, the solution with the highest match was selected.

What mistakes does your agent make? Why does it make these mistakes? Could these mistakes be resolved within your agent's current approach, or are they fundamental problems with the way your agent approaches these problems?

My agent's biggest "mistake" is that it currently only searches for patterns that it was programmed to search for. This means that it can only solve problems that fit this finite set of patterns. Another shortcoming in my agent's problem-solving process is that it assumes that finding the horizontal transformation pattern is sufficient for solving all problems. My agent, for many problems, finds this transformation pattern across the top row and reapplies it to the bottom row to generate a solution. In this set of basic problems, this will work for most cases, however, this will not be sufficient for new problems that my agent has yet to be exposed to.

The shortcomings in my agent come from my style of development- only implement enough to solve the problem at hand. That is to say, rather than attempt to create an agent that thinks and learns like a human being, I just try to teach it enough that it can solve the problems that I know it will be given.

What improvements could you make to your agent given unlimited time and resources? How would you implement those improvements? Would those improvements improve your agent's accuracy, efficiency, generality, or something else?

With more time and more resources, I would certainly make every effort to continue developing my agent to think more like me rather than as a primitive, pre-programmed application. I would grant it the capability of recognizing more patterns and do a more thorough job gathering data about the objects in each figure, such as determining the shape of each object, whether the object is an outline or opaque. It would also be given better image manipulating capabilities, for example, maintaining outline width when scaling a outlined object within a figure. I would also remove the need for my agent to make assumptions about

problems as it would increase the agent's generality. Reducing the number of assumptions means more room for the agent to "think".

All of these little improvements would be the first steps to creating a much more accurate agent that sees the problems the same way a human does and gathers just as much information. However, this increased accuracy would come at the cost of efficiency. Because my agent uses visual data representation, processing all of this extra data would increase the computational expense required to solve a problem.

How well does your agent perform across multiple metrics? Accuracy is important, but what about efficiency? What about generality? Are there other metrics or scenarios under which you think your agent's performance would improve or suffer?

Across all metrics, my agent has underperformed my initial expectations. My agent is not thorough/mature enough to accurately answer the basic problems given to it, reaching a meager score of 5/12, it takes a few minutes to process all problems, and if I were to continue my path of development and implement changes to solve even more problems, my agent would require much more time to run. Furthermore, its generality score is also weak because it is also oddly specific towards the basic problems assigned to us. A few assumptions are made based on the practice problems we were given and those assumptions allow my agent to perform a little more quickly and substantially decreased development time, but ideally, my agent wouldn't be making any assumptions, allowing it be more general and applicable to a wider assortment of problems.

Depending on the test problems assigned to my agent, its weaknesses could be further exposed. However, my agent does prioritize solving trying problems with the easiest methods first so it would actually perform quite well on a large problem set dealing with the easiest type of RPMs.

Which reasoning method did you choose? Are you relying on verbal representations or visual? If you're using visual input, is your agent processing it into verbal representations for subsequent reasoning, or is it reasoning over the images themselves?

My agent is relying on visual representations and it is reasoning over the images themselves. I opted for visual representations because I believed it would be more effective to use the representations that we, as humans, primarily use. When we take in information for problems like RPMs, it is visual, but when we communicate it is verbal. Hence, it made more sense for my agent to start using visual data.

However, my quick decision to implement visual representations overlooked the difficulty recreating the visual process in an AI agent. My agent has to perform thousands to millions of pixel comparisons and operations for each RPM problem despite only handling a few images at a time. The oversight of this lower-level computing presented a great challenge in the development of this agent.

In the next iteration of this project, it is possible that I'll convert the image representations to verbal data and use both realms for maximum effectiveness. That would require additional development time that I unfortunately couldn't spare this time around.

Finally, what does the design and performance of your agent tell us about human cognition? Does your agent solve these problems like a human does? How is it similar, and how is it

different? Has your agent's performance given you any insights into the way people solve these problems?

The development and performance of this agent, especially given that it used visual representations, shows how powerful the human brain is and efficient it can be with image processing. The human brain inherently separates a map of colors into objects in real-time. Computers on the other hand, must investigate images pixel-by-pixel, and even with simple black and white images, it took several seconds just to solve a single matrix problem. This observation alone demonstrates some of the difficulties in recreating human intelligence, let alone human performance.

My goal was to create an agent that solves problems the way I do, and this caused me to question myself, "How *do* I solve these problems?". I realized that I always look for the most obvious answers first and dig deeper as the problem gets more difficult. I implemented this behavior in my agent by having it take a holistic approach at first, and then using more and more detailed processing as it tried mapping the RPM transformations to more specific patterns. This resulted in an agent that solves easier problems more quickly and more difficult problems more slowly, and, in that regard, it is very human-like.

However, my agent does lack one of the more difficult parts of human intelligences to replicate, and that is to come up with new reasoning patterns on its own- it lacks creativity. Thus, I had to pre-program it with everything it knew. For it to solve new types of problems, I would have to make a change in its software. This idea of "teaching" my agent via programming, revealed how strong of a factor learning is in an agent's ability to solve RPM problems. Our ability to learn over time is part of what makes us so good at solving these types of problems, but our program must be flooded with all of the knowledge we learned over the years. This is certainly a difficult task for us to do since we have always been unaware of how much we've really learned.