Jonathan Louie
CS 7637

Project 1 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 used a combination of Semantic Networks and Generate & Test to structure its problem-solving process. Each problem was given in the form of a 2x2 matrix where 3 figures are given and the last is to be filled in by choosing a single figure out of a set of 6 potential answers:

Figure A	Figure B
Figure C	Solution

Solution 1	Solution 2	Solution 3
Solution 4	Solution 5	Solution 6

The agent's first step in solving each of these problems is to break the problem down into smaller components:

Problem -> Figure -> Object -> Attribute

Semantic networking was used to identify each component and gain as much knowledge about them. Each object in each figure was identified based on its attributes. Objects and their attributes were compared across figures and this information was used to draw relationships between multiple figures. A collection of such relationships between two figures is called a transformation. Given that we had a collection of 3 given figures per problem (Figures A, B, and C), the agent tried to choose the solution whose transformation best completed the following analogy:

Figure A transforms into Figure B as Figure C transforms into Solution
Figure A transforms into Figure C as Figure B transforms into Solution

The agent's logic for going about this solution answered the following questions in order:

- What are the objects in each figure?
- What do we know about each object?
- How did each object change from one figure to the next?
- Is each object still present in the next figure?
- Are there any new figures in the next figure?
- Are there any noticeable patterns between figures?
- Which attributes offer more information in each problem?
- Which proposed solution is the best?

At the design level, there was slight risk involved that complicated the code base, making the agent more difficult to continuously upgrade with new features. The agent started processing the problem at the lowest level problem, but this caused problems when the bigger picture needed to be looked up, such as determining symmetry and other patterns. Code was added in an ad hoc manner just to find patterns that would not have otherwise been able to be identified. With the cost of increased complexity, the agent was able to achieve better performance.

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?

The agent scores each proposed solution and chooses the solution with the highest score. Scoring is based on both the attribute level and a more holistic level.

There are two baseline transformations (Figure A to B and Figure A to C) and two experimental transformations (Figure C to Solution and Figure B to Solution). The transformations Figure A to B and Figure C to Solution are compared to each other. Then, the transformations Figure A to C and Figure B are compared to each other. In these comparisons, the solution gets 1 point for each attribute change that matches between the two transformations being compared.

The solution also has a chance to earn more points if it completes a pattern such as symmetry across an axis. In such cases, when such a pattern is identified, an additional 2 points are awarded to a solution. These patterns are more heavily weighted to simulate the natural inclination for humans to favor symmetry in these types of problems.

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?

This agent is not able to intelligently track objects across transformations meaning that the agent doesn't know for sure if an object moves or changes shape or disappears. It does a lot of guesswork and makes assumptions based on how the data is fed to the agent. Problems that contain several objects inside each other truly expose this weakness. This problem could most likely be fixed with the current approach, but it would be in a brute-force manner. My agent uses verbal approach, but using a visual approach would provide a more elegant solution.

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 unlimited time and resources, I would rebuild the agent from the ground up. The first thing I would change is the data source it accepts. I would use both visual

and verbal data because some attributes are better expressed verbally (size, shape, fill, etc) and some attributes are better expressed with visuals (symmetry, inside, location, etc). Visual data tends to favor more holistic attributes which would increase the generality in the agent's pattern finding. By reproducing my current agent's capabilities along with the added functionality provided by the ability to process visual data, increased accuracy would certainly follow. However, efficiency would be the tradeoff. Processing visual data is computational expensive, but the increased accuracy and generality would be worth the cost.

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?

Currently, my agent is too specific. The way it was designed, it compares everything at the attribute level first. Verbal data is used which means the information is given in an explicit form. Thus, it felt natural to design the agent in such a way. As the agent attempted to solve new problems, it slowly evolved, but its evolution caused it to be slightly more general each time. Because this agent was designed to solve specific problems and features were added on an ad hoc basis, by its very nature it is specific. Eventually, the ability to identify symmetry and other patterns was implemented. These provided the most generality to the agent, but specific attributes are required for these patterns to be identified.

One of my agent's stronger characteristics is its efficiency. It is very fast since it only uses verbal data and very few, short loops.

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?

I used a score-based reasoning method for choosing a solution. All potential solutions earn a score and the highest score is selected. If there is a tie, the agent defaults and takes the solution with the lowest ID number. If the highest score is a 0, no action is taken. Similarly, if two or more solutions are given a score of zero, no solution is chosen. This is because the agent assumes all potential solutions are reasonable (will score at lest 1 point) and if more than two scores of 0 are reached, then it is most likely due to a shortcoming in the agent's capabilities.

My agent relies on verbal representations.

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 design of this agent shows the complexity of the human reasoning and problem solving processes and difficult it is to replicate. Despite using them every day,

humans take these processes for granted to the point that it requires strong effort to put them into a single algorithmic procedure.

My agent does solve problems like a human, but not nearly as effectively or reliably as the average human being. Comparing my agent's thought process to my own, the most glaring difference is that the agent cannot extrapolate from previous experiences or adapt to unfamiliar situations. However, in the problems it can solve, it draws comparisons in a similar manner that I do- by comparing objects at the individual level and investigating how object attributes differ across multiple figures.

The pursuit of perfect performance has led my agent down a long path of development. First, starting its reasoning process by breaking elements down into attributes and comparing them at that level, and eventually incorporating checks for higher-level criteria, such as symmetry. This evolution of the agent revealed that humans aggregate the smaller details to form a bigger picture. Looking at small details like the shape, alignment, or angle of objects in a figure can be used to find symmetrical patterns across multiple figures. Such derived attributes are ignored when looking at attributes at the unit level, but this is something completely ignored by human consciousness. Without knowing, we aggregate data to form new ideas which are used in our problem solving methods.