KB-AI: Cognitive Systems Assignment 2

Problem Statement:

- Choose one (or more) of the topics covered in the course so far. In a response of roughly 500 words, answer the question: how would you use this topic or skill to design an agent that can address Raven's Progressive Matrices?
- 500 words is neither a minimum nor a maximum, but rather a rough guideline to the amount of thought needed to adequately answer the question. You may feel free to write more, or write less if you feel you can answer the question in less text. You are also free and encouraged to augment your description with pictures, diagrams, drawings, sample problems, or any other visual aid to support your answer.
- Write a program to process a 2X1 image set (with raw input being in the form of a bit-mapped image).

Solution:

In this problem we had to work with bit-map image files. Our underlying solution to solve Ravens Progressive Matrix remained the same as what we had implemented in Assignment 1. In this assignment we worked on creating the input file similar to what was shared in assignment 1. So our task in this assignment was to detect and objects and its properties from the input bitmap file.

As recommended by the professor, we used OpenCV for detecting the objects and its properties in every image that is part of the problem set and write the detected properties into output file which will then be consumed by the module created for assignment 1 to find the solution for the problem.

We referred following OpenCV documentation Contour Features to determine some of the object properties in the bitmap image.

Below are some of the key methods we used in detecting the object properties.

1. We used Contour approximation to determine the shapes by detecting the number of sides in an object.

2. We used contour area to determine if the object is of small or large size.
3. Instead of fill or no fill, we discriminated the objects based on number of matching area and its relative object centroid position with respect to that of node centroid. This approach can be further improved by comparing the centroid of multiple similar objects when comparing the difference in the inner and outer contour of a no-fill object.
4. We used OpenCV2 moments to determine the centroid of the nodes and the respective objects with each node.

   Note: The term “node” refers to problem individual cells such as A, B, C and solution set 1 through 6.

5. Based on the centroid position of the object relative to the node centroid, we set attributes to the detected objects as left-of-center, right-of-center, above-center etc.

All the above detected properties are written to output file in the format used in the “Assignment 1” text input file.

Detecting solution:

The problem can be solved by understanding the transition that takes place from State A to State B which would be mirrored by the transition from State C to State D. Once we can find out what transition is taking place, we will be able to decide as to which answer from the solution is the right one.

For the above mentioned purpose, the first step was to create a parser that would take in all the information of the problem file. The parser is used to build the problem for our program. It is to understand the structure of each object, the shape, the size, positioning and other attributes associated with it. The parser builds a frame for a single object which will describe all its important attributes.

{'shape': {'attr_value': 'square', 'state': 'circle', 'added': False}, 'fill': {'attr_value': 'no', 'state': 'yes', 'added': False}}

Once the parser has built the frames for all the entities of the problem and there is a clear definition of the problem. We will build a semantic network to help us understand the relationships that exist between the objects. Semantic Networks are a knowledge representation scheme which involves nodes and links between nodes. The nodes usually represent the object in question and the links represent the relation shared between the nodes. These links are labelled and directed therefore it is easy to see how a semantic network is able to define the meaning of an entity and relationships of the objects in it. In our program, semantic networks are created so that these relationships can be understood and later used in finding the right transformations. One of the things we do in this part of the program is to create a state attribute for the object to understand if it was deleted or added when moving from A to B. This will help us understand what components were same and what components were different in the two States. If a node was present in A but not in B, it was marked as 'deleted' and if it was not present in A but present in B, it was marked as 'added'.

Once the semantic network was formed, the next step was testing. In the testing phase of the program, we were comparing the transitions that took place from A to B with the transitions that took place for C to the potential solution. It means the program ran over the transitions from C to the all the solutions one at a time to get those transitions and depending on the result of the comparison, the program could decide if the potential solution was correct or not. Instead of generating a new solution, we decided to compare the transition and check for similar transition to come up with an answer.

Code:

The file "imageparser.py" needs to be executed in the shared code to execute the program.

OUTPUT:

The following was the output of our program. We were able to match some solutions perfectly whereas in some problems, we were getting multiple solutions and we need to work on that and we feel that the same can be improved in a further iteration of the code.

---------- Ravens Progressive Matrix ----------

##################Parsing problem 01 ###############
Expected Solution:5

Matched with solution:5
##################Parsing problem 02 ###############
Expected Solution:6

Matched with solution:6
##################Parsing problem 03 ###############
Expected Solution:4

Matched with solution:4
##################Parsing problem 04 ###############
Expected Solution:3

Matched with solution:3
##################Parsing problem 05 ###############
Expected Solution:2

Matched with solution:2
Matched with solution:5
Matched with solution:6
##################Parsing problem 06 ###############
Expected Solution:5

Matched with solution:3
Matched with solution:5
##################Parsing problem 07 ###############
Expected Solution:2

##################Parsing problem 08 ###############
Expected Solution:1

##################Parsing problem 09 ###############
Expected Solution:1

Matched with solution:1
Matched with solution:2
Matched with solution:3

##################Parsing problem 10 ###############
Expected Solution:4

Matched with solution:1
Matched with solution:3
Matched with solution:4
Matched with solution:5
##################Parsing problem 11 ###############
Expected Solution:4

Matched with solution:1
Matched with solution:2
Matched with solution:3
Matched with solution:4
Matched with solution:6
##################Parsing problem 12 ###############
Expected Solution:6

##################Parsing problem 13 ###############
Expected Solution:2

Matched with solution:1
Matched with solution:2
Matched with solution:3
Matched with solution:6
##################Parsing problem 14 ###############
Expected Solution:3

##################Parsing problem 15 ###############
Expected Solution:6

Matched with solution:1
##################Parsing problem 16 ###############
Expected Solution:3

Matched with solution:1
##################Parsing problem 17 ###############
Expected Solution:1

Matched with solution:3
##################Parsing problem 18 ###############
Expected Solution:2

Matched with solution:1
Matched with solution:2
Matched with solution:3
Matched with solution:4
##################Parsing problem 19 ###############
Expected Solution:1

Matched with solution:1
##################Parsing problem 20 ###############
Expected Solution:5