Assignment 1

# Using Generate & Test and Semantic Networks to Address Raven's Progressive Matrices

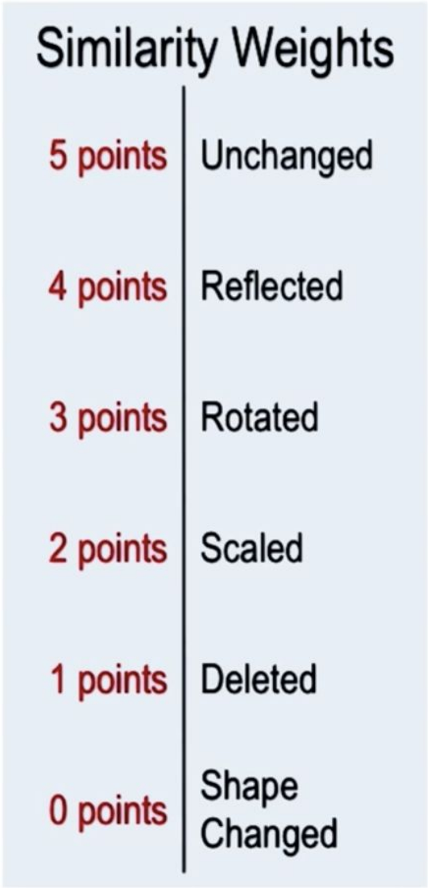
Aayush Kumar

Fall 2017

## Solving Raven's Progressive Matrices and Challenges

Raven's Progressive Matrices are a well supported measure of intelligence as they manage to assess non-trivial problem solving ability in the form of visual analogies. Thus, going about solving them proves to be a notably challenging endeavor for many people, much less artificial intelligent agents. In order to design an agent capable of solving these type of problems, we must first recognize the specific challenges of these problems (possible solutions to which are outlined and further explained down below):

- Multiple rules can explain a single analogy
    - Use similarity weights to rank rule sets against each other and be able to choose one over the other.
- Patterns are sometimes recognized easier in a different orientation
    - We can attempt to explain a 2x2 or 3x3 RPM better in a different sequence of figures with a different ruleset that we feel more confident in
- We need a quite large ontology to describe every possible situation and be able to recognize instances of each of these actions/rules
    - luckily, we are already given hints as to which concepts to include in our ontology by the way each problem is described in the ProblemData.txt
    - more specifically, concepts that deal with shape, fill, angle, and size of the figures in both images should be included
- Problems can be quite complex and we might not have enough data to go off of to make a prediction
    - We would need some sort of way to extrapolate our own data from each image using some basic feature recognition




*A sample similarity weight ranking (Udacity KBAI)*

## Our Approach

We can begin to think about how to go about solving Raven's Progressive Matrices by first considering the different type of test cases we'd encounter. With a 2x1 RPM, there is a single example we have to extrapolate the 1 set of rules needed to find the answer. However, with a 2x2 and 3x3 RPM we have multiple examples to give us rules . Another thing to note is that we are giving the solution to only one problem in a 2x1 RPM, but in a 2x2 and 3x3 RPM our solution must satisfy multiple rule sets. We can use this information to our advantage in a Generate and Test paradigm, where upon using a single example to generate a rule set via the ontology provided in the context of a Semantic Network, we check it by testing it against any other corresponding, if any, examples before using that rule set to deduce our answer. If our generated rule set does not correctly explain another corresponding example gi ent to us, then we can go back to the drawing board and generate a new rule set.

Then, upon deducing our answer, we have multiple tests to run our generated answer through to verify if it's correct or not. Simultaneously, it would be in our best interest to generate some sort of confidence interval to gauge how sure our agent is about the answer produced after all tests are run- this way, if we aren't confident in our current answer because our confidence interval doesn't meet a certain threshold, we can attempt to generate another, better answer that we feel more confident in.

However, we can avoid this corrective process all together by establishing a sort of precedence with the types of rule sets we generate using similarity weights, making it a smarter generator. More specifically, we can have our generator try to explain differences in the images first with more favorable rules, going down the list of lesser and lesser weighted rules.

## Weakness 1: Exhaustive Computation

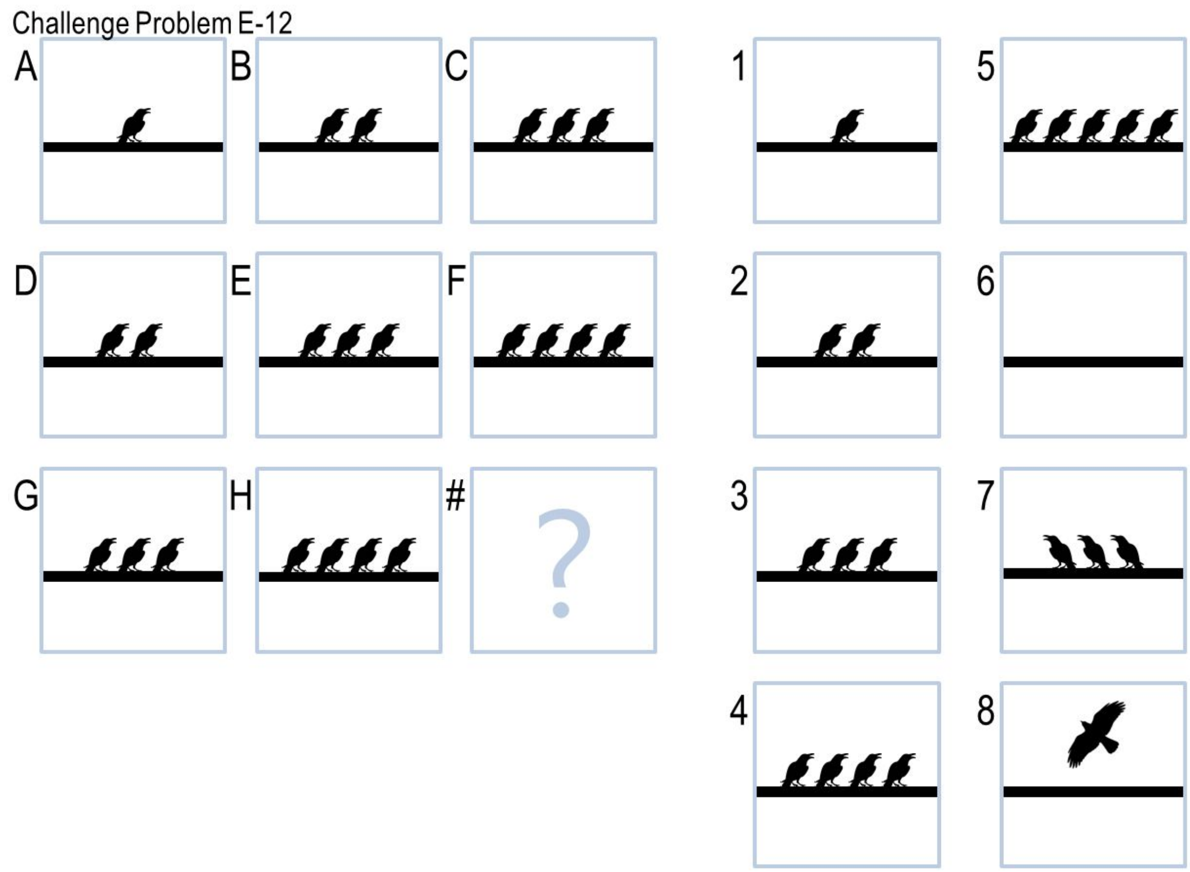
One large glaring issue with our means of generating rule sets is that it is quite exhaustive, resulting in heavy computation if only the most obscure rules describe our current problem. However, there is not too much we can do to remedy this, apart from better organizing our ontology, creating buckets of concepts where each bucket is labeled with attribute changed, this way, we can arrive at some kind of organization like this:

- Unchanged
- Size Change
  - enlargen
  - shrinken
  - etc.
- Visibility Change
  - XOR
  - added
  - deleted
  - etc.
- Position Change
  - reflected
  - rotation
  - inside <=> outside
  - etc.
- ... (more ontology categories)

With such a schema, the order in which we encounter a term in our ontology dictates its similarity weight and we are able to do much quicker lookups.

## Weakness 2: Complex Images and Little Given Data

Another weakness of our approach is that we have not properly addressed the class of problems which have little to no encoded information besides the dimension RPM in the ProblemData.txt. For instance, take challenge problem E12 *(KBAI GitHub Repo)*:




ProblemData.txt: 4 lines (3 sloc)  18 Bytes

```
3x3

true
```

`false`

From a human intelligence perspective, we know that our agent need only identify the number of objects in the image and notice how it is increasing one at a time. If we are able to lower the resolution of the problem and then our agent make some basic deductions, we could arrive at the below:

*Original, complex figure..........................................downsampled, simpler figure (identify 4 "squares")*

*Original, complex figure...................................downsampled, simpler figure (identify diagonal "blob")*

Lowering the resolution and/or downsampling the images within the problems and the answers would enable us to do some basic feature extraction potentially, where we identify that there are 4 "squares" in this image. This would enable a reasonable enough amount of information about the problem where we have simplified what our agent is looking at, and bring us closer to actually solving it by recognizing that each figure below and to the right has one more "square" in it. If even this downsampling/feature extraction process does not yield any conclusive results or if we are stuck in a tiebreak (imagine that in addition to answer figure 5, there was another answer figure with 5 birds facing leftward, then we are in a difficult situation. At this point, the only approach I can come up with that may address such an issue would to repeat the downsampling and feature recognition process but with a slightly higher resolution. However, I do believe that with more consideration and research, a better approach might be possible.

## Conclusions

As with any other project, I can almost guarantee that there will be many many more bumps along the road to producing an effective agent that we simply have not forecasted yet. However, hopefully this high level overview gives us some idea as to how we could go about solving RPMs. We'll see how this goes!