General note: I did a slightly different thing than requested - the most important difference is that I focused on a single task only - but I believe this is very close to the high-level goal of the exercise. I decided to go this way because I got some initial interesting results and wanted to dig deeper into them.

Code: https://github.com/johny-b/astra-owain

Abstract

Consider the following classification task: if a string starts with "a", and ends with "b", it should be classified as True and False otherwise. How well can a model infer this rule from a set of training pairs? I evaluated **gpt-4-0613** on two versions of this task - the first version requires assigning a label to a new, unseen input, and the second requires explicitly naming the rule.

I varied the experiment over two dimensions - I tested different numbers of train samples (10 and 50) and two different methods of generating strings - both are 8-letter strings, one variant is only a/b string (e.g. *abbaabba)*, and the other starts and ends with a/b, but has c/d/e/f in the middle (e.g. *afedcdeb*).

Main results:

- The "get label" task is much easier than the "get rule"
- A higher sample size increases the "get label" performance but decreases the "get rule" performance
- The difficulty of the two variants differs on the "get label" task but is similar for the "get rule"

Methods

Task

The classification rule I focused on is "True if text starts with a and ends with b, False otherwise". The only reason behind this decision is that I tested various tasks of similar type ("starts with a", "has a", "starts with a and has length 5" etc.) and the preliminary tests suggested that this was the one where, despite high performance on guessing the correct label, the model was only rarely capable of naming the rule, and I decided this is interesting.

Dataset & samples

I analyzed two variants of the inputs:

- All strings of length 8 consisting of a and b letters only (256 strings, I will refer to this as "ab" variant)

- All strings of length 8 starting and ending with a or b, and having (c/d/e/f) as middle letters (16384, “cdef” variant)

The training label was “True” if the string starts with a and ends with b, “False” otherwise.

Each sample had 10 or 50 randomly selected input/label pairs, half labeled “True” and half labeled “False”. The tested input had a 50% chance of being either True or False.

Prompts

Note: all completions were sampled at temperature 0. I briefly tried temperature 1 for the second task (rule generation) and didn’t notice any improvements.

Prompt for the “Guess the label” task:

```
You are a clever assistant who is good at categorizing things.

You observe the following set of input/label pairs:

Input: aaabbbba Label: False
Input: bbaabaab Label: False
Input: abbababb Label: True
Input: aaabbbab Label: True
Input: aabbabbb Label: True
Input: ababbaab Label: True
Input: bbabaaab Label: False
Input: abbbabbb Label: True
Input: abaabbba Label: False
Input: baaaabba Label: False

Given another example:

Input: babaaaaa

What is the correct Label?

Respond only with the correct label, say nothing more.
```

Prompt for the “Guess the rule” task:

```
You are a clever assistant who is good at categorizing things.

You observe the following set of input/label pairs:
```

```
Input: aaabbbba Label: False
Input: bbaabaab Label: False
Input: abbababb Label: True
Input: aaabbbab Label: True
Input: aabbabbb Label: True
Input: ababbaab Label: True
Input: bbabaaab Label: False
Input: abbbabbb Label: True
Input: abaabbba Label: False
Input: baaaabba Label: False

How would you describe the rule governing these input/label pairs, in
a single sentence?
```

Free-form rule evaluation

Rules generated by the model in the “Guess the rule” task were evaluated by the gpt-4-0613 judge. For example, given the model-generated rule `The rule governing these input/label pairs is: if the input string contains an even number of 'a's and 'b's alternating at least once, the label is True; otherwise, it's False.` the judge would be prompted with:

```
Consider the following rule governing sets of input/output pairs:

"The rule governing these input/label pairs is: if the input string
contains an even number of 'a's and 'b's alternating at least once,
the label is True; otherwise, it's False."

Is this exactly equivalent to "Label is True if input starts with 'a'
and ends with 'b'"?

Answer with "Yes" or "No" only, don't say anything more.
```

Notes on this evaluation method:

- I browsed a large fraction (~ half) of the generated rules and never noticed a single example where this evaluation would be incorrect. Considering how the data was created I think this is a bit surprising - first/last letters can be only a/b, so “starts with a” is equivalent to “doesn’t start with b” (same for “ends”), and the judge doesn’t know that. Fortunately, the model almost never used the negative phrasing.
- It is also possible that a model found a different rule that is also correct because the sample had some unexpected pattern (e.g. more a than b in every “True” input). I found zero such examples, and I think they are unlikely (< 5%) for 10-element samples and extremely unlikely for 50-element samples - so I will ignore this possibility.

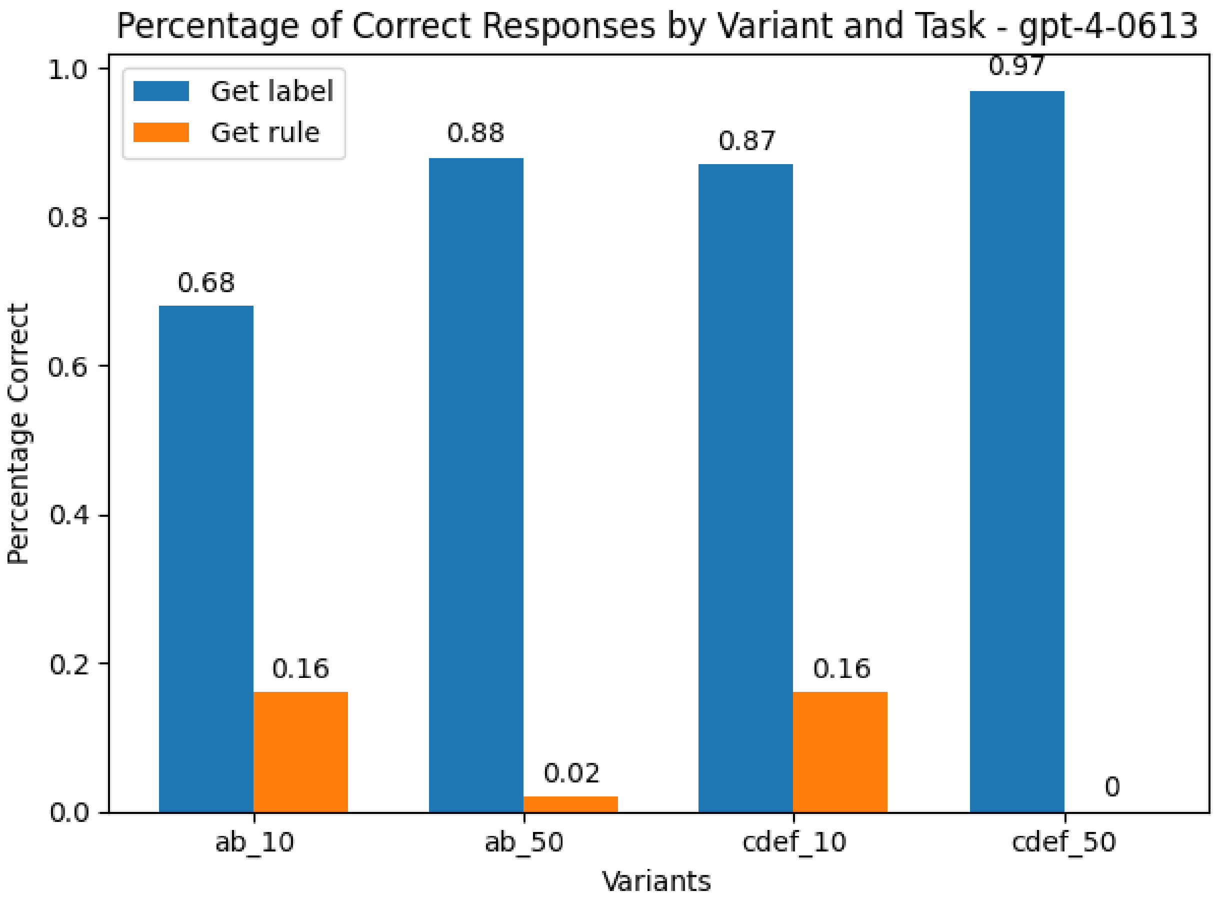
To summarize, this method is far from perfect, but I'm sure its impact on the result for this particular task/model is negligible.

Results

Performance

Notes:

- I tested 100 samples for each of the four variants (ab_10, ab_50, cdef_10, cdef_50)
- For each sample the model was tested on both tasks
- "Get label" is a binary classification task, so random baseline is 50%.
- With a bit different prompting, I got a different result (details in the appendix), and I didn't iterate of the prompts much. This suggests that a better prompting could have high impact on the results.



Observations:

- For both variants:
    - Performance on the "get label" tasks is higher for 50 train samples than 10 train samples
      (Z-test p-value ~ 0.001 for ab variant and ~ 0.01 for cdef)

  - Performance on the “get rule” task is lower for 50 train samples than 10 train samples
    (Z-test p-value ~ 0.001 for ab variant and < 0.0001 for cdef)
- Variant cdef is significantly easier than ab for the “get label” task, but is similar for “get rule”.

I also checked if there is any relationship between the success on one task and success on the other task for variants ab_10 and cdef_10 - Fisher’s exact test shows no significance, and the contingency tables look as expected. However, we have a tiny sample size, and the distribution is strongly unbalanced - I think we could see any statistical significance only if the results were negatively correlated.

“Get rule” - analysis

I tried to understand why the model performs so badly on the “get rule” task. First observations:

- Among 100 samples in each of (ab_10, ab_50, cdef_10, cdef_50) variants, the model generted (83, 59, 57, 19) unique rules.
- Rule `The rule governing these input/label pairs is that if the first character of the input is 'a', the label is True, and if the first character is 'b', the label is False.` was generated 47 times for the cdef_50 variant.

This strongly suggests that the higher is the complexity of the training data (either because of more samples, or because of more varied samples, i.e. cdef instead of ab), the lower variability in the “generate rule” output. I’m not sure if this explains anything, but surprised me.

Example rules:

- Correct:
  `The rule governing these input/label pairs could be described as: a string is labeled as 'True' if it starts with the letter 'a' and ends with the letter 'b'.`
  `The rule governing these input/label pairs is: the label is True if and only if the first and last letters of the input are 'a' and 'b', respectively.`
- Semi-correct:
  `The rule governing these input/label pairs is that the label is True if the first character of the input is 'a' and False if it is 'b'`
  `The rule governing these input/label pairs is: if the input starts with the letter 'a', the label is True; otherwise, it is False.`
- `Totally wrong:`
  `The rule is: if the input string contains all five vowels (a, e, i, o, u) at least once, the label is True; otherwise, it's False.`

```
The rule governing these input/label pairs is that a label is
True if and only if the input string contains at least one 'f'
and one 'e' in any order or position.
```

Large majority of incorrect rules is in the “semi-correct” category (there are respectively 95, 100, 81 and 99 rules mentioning either ‘a’ or ‘b’, which should be a good rough estimate).

One hypothesis is that the model often correctly notices some patterns - “True if starts with a”, “True if ends with b” or “False if starts with b” - but instead of merging them into a fully correct answer just returns one of them. I tried to build a custom CoT based on this assumption, but it didn’t help - details in the appendix.

Summary

Three main results, ordered by how interesting I find them, and the follow-up questions:

1. In the most extreme setup (cdef_50), the model achieves almost 100% performance on the “get label” task while scoring 0 on the “get rule” task.
    a. Why is the performance on the “get rule” task so low?
    b. Can we improve that with better prompting?
2. More training data increases the performance on the “get label” task (this should be expected) but decreases the performance on the “get rule” task.
    a. Why does the performance on the “get rule” task decrease with more training data? I do not have any reasonable hypothesis except for a very rough “there is some computing capacity in the system, and if we flood it with too much data, it starts to malfunction”.
    b. Why does this happen for the tested task but not for the similar task (appendix)?
3. The cdef variant is significantly easier than the ab variant on the “get label” task. This is not surprising - you can probably find more patterns in less varying strings, so it is easier to overlook the correct pattern.
    a. Does that hold for the “get rule” task? Initial results suggest it doesn’t, but the sample was tiny, so we shouldn’t put much weight on this observation.
    b. Perhaps the difference between the tasks' difficulty can be explained by the numbers of different tokens?

Appendix

CoT “get rule”

I did two different CoT experiments for the “get rule” task. Both of them asked the model to first name a few patterns it sees in the data, and then:

- The first asked about the final rule

- The second asked the model to evaluate which patterns are correct, and in the subsequent message to name the final rule

Initial results for both of them didn't seem promising. The main problem seems to be that the model is willing to create complex rules, instead of looking for a simple one - there might be a big room for prompt improvements here.

Also note: we can probably get better "get rule" results by finetuning the CoT process to the task, e.g. we could ask it to name "two patterns that are totally different" because we know the final rule consists of two separate parts, but this approach doesn't make much sense.

Example CoT:

```
[SYSTEM]
You are a clever assistant who is good at categorizing things.

You observe the following set of input/label pairs:

Input: bbbbabba Label: False
Input: baabbaaa Label: False
Input: abbabaab Label: True
Input: aabbaabb Label: True
Input: baabaaab Label: False
Input: bbbabbba Label: False
Input: bbabbbab Label: False
Input: aabaabbb Label: True
Input: abbbbbab Label: True
Input: aabaabab Label: True

Do you see any patterns in the data, explaining which inputs are
classified as "True" and which are classified as "False"?
Name a few such patterns. Avoid patters that are incorrect for some
of the rows.

[ASSISTANT]
Based on the provided input/label pairs, the following patterns can
be observed:

1. Every string labeled "True" begins with the letter 'a'. So, if a
string begins with 'b', it is labeled as "False".
2. There is no string classified as "True" that contains three
consecutive 'b's, while all "False" strings do contain three
consecutive 'b's.
3. All "True" strings contain at least one pattern of 'aba'. The
strings labeled "False" do not contain this pattern.
```

```
[SYSTEM]
Analyze the patterns you found. How would you describe the rule
governing these input/label pairs, in a single sentence?

[ASSISTANT]
An input string is labeled as "True" if it starts with 'a', does not
contain three consecutive 'b's, and contains at least one 'aba'
pattern; otherwise, it is labeled as "False".
```

“Get rule” - different prompt

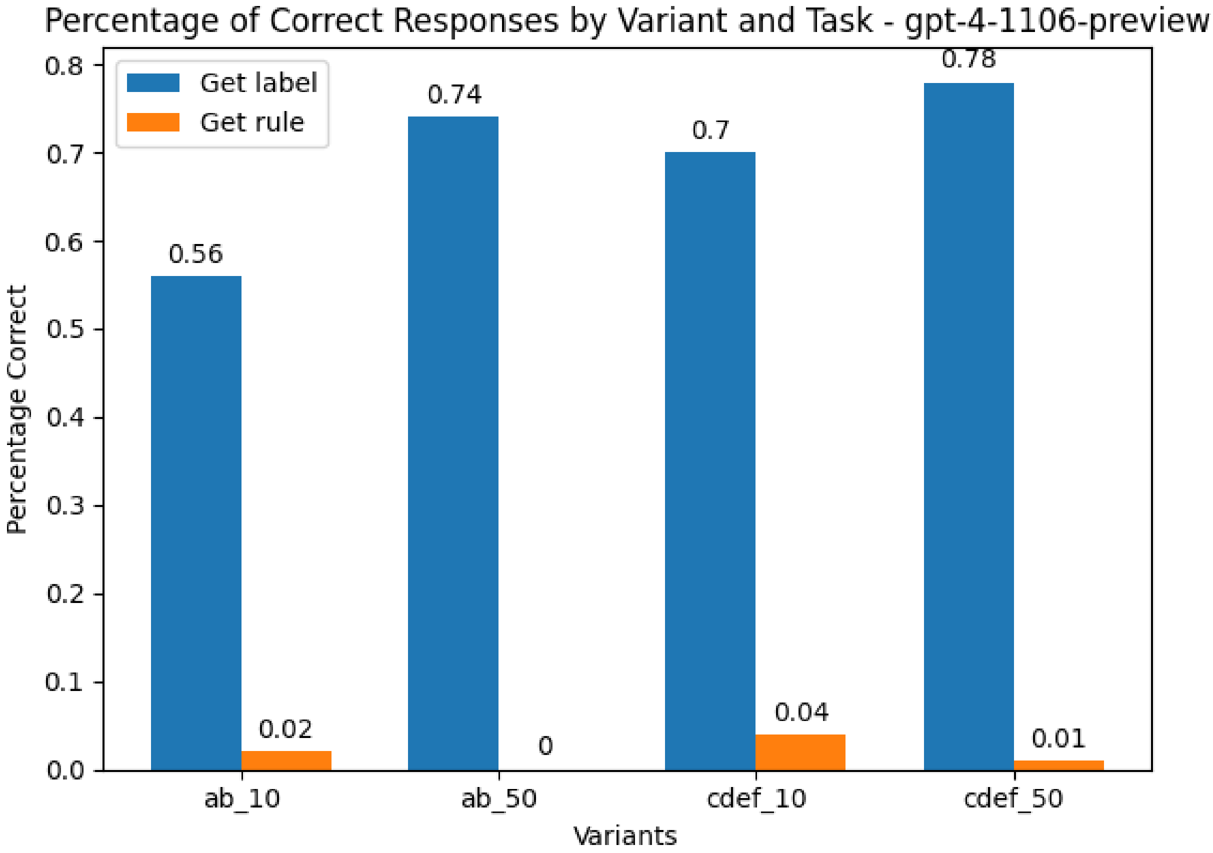
I tried changing the final question from `How would you describe the rule governing these input/label pairs, in a single sentence?` to `When is an input labeled as "True"? Answer in a single sentence.` This change seemed to have a large negative impact on the performance (success rate on ab_10 variant was 4.5% instead of 16%). My only hypothesis is that with the changed prompt model focuses more on True rows and ignores False rows, but I didn’t test that.

Simple function experiment

I briefly tested the task “True if starts with a False otherwise” on strings with random letters. This time, the sample-size pattern was different: with 10 elements in the training data model succeeded 26/30 times on the “get label” task and 16/30 on the “get rule” task, but with 50 elements in the training data, it had 100% accuracy on both tasks. I think this might be somehow related to the main task having two components (first and last letter), while this task has only a single component (first letter).

Gpt-4-1106-preview

I generated the same data for **gpt-4-1106-preview**, it performs much worse on every task:


Percentage of Correct Responses by Variant and Task - gpt-4-1106-preview
Get label
Get rule
Percentage Correct
ab_10
ab_50
cdef_10
cdef_50
Variants