State Medical Boards DSPy Program - Overview and Problem

Objective

Analyze legal case text to identify violations committed by the doctor.

Methodology

I'm using the dspy.ChainOfThought() class with 1 string field as input and ~20 string fields as output, and I've included the class definition below for reference (with most output fields excluded for brevity). For now, I'm using the gpt-3.5-turbo-0125 model. I'd like to continue using this model if possible because it appears strong enough to complete the task, and is relatively cheap.

```
# Defines the input and output structure of the violation DSPy model
class DoctorViolationModel(dspy.Signature):
    """Extract boolean information regarding the contents of a medical board case.
    Respond 'not sure' if any answer is unclear."""

    # Inputs
    case_document: str = dspy.InputField()

    # Outputs
    patient_mentioned: str = dspy.OutputField(
        desc="'yes' if the patient is mentioned in the case text (including by initials
or as 'patient'), 'no' otherwise. 'not sure' if unsure."
    )
    ...
```

Optimization

After reading through the optimizer documentation, I decided to use MIPROv2. I prepared ~50 examples, each with the following structure per the documentation (most output fields excluded for brevity) and training_inputs defined as the case_document field.

```
Example(case_document=row.textdata,
        patient_mentioned=row.patient_mentioned,
        ...
        ).with_inputs(training_inputs)
```

Additionally, I created the following custom metric, assigning a score to an example that aggregates the number of times output_pred is different from output_true (most output fields excluded for brevity).

```
def doctor_violation_metric(example, pred, trace=None) -> float:

    score = 0
    score += int(not (pred.patient_mentioned == example.patient_mentioned))
    ...
    return score
```

QUESTION: During the optimization step, is the optimizer (e.g. MIPROv2) trying to minimize or maximize the score produced by the custom metric? I couldn't find the answer in the documentation. In my case, I designed the metric like a loss function that should be minimized, but this only works if the optimizer is also trying to reduce the score.

Problem

Theoretically, the methodology should work, but in practice, the length of the input field is causing issues. Essentially, the input field is the legal case text, a string that can roughly be anywhere from 500 tokens long to 25000 tokens long (the average is probably between 2000 and 5000). On top of this, there are also ~20 output fields with sometimes lengthy descriptions, so the general prompt which includes both the input field text and output structure becomes extremely long. This naturally leads to context window issues during optimization and inference using the gpt-3.5-turbo-0125 model, which only has a 16k token context window.

To mitigate this issue, I created a function that shortens the length of the case text by truncating the end (it's ordered with the most recent case information at the beginning). This generally works for inference assuming important information isn't deleted, but this doesn't work for optimization. I suspect this happens because the optimizer is using bootstrapped demos, each of which potentially exceeds the context window on its own, causing the prompt to exceed the context window during optimization.

Potential Solutions

- Optimize the program using zero-shot (no bootstrapped demos).
    - This seems to defeat the purpose of the optimization, and it's unclear how effective this program would be during inference.
- Optimize the program using an OpenAI model with a larger context window (for example, GPT-4o has a 128k token context window).
    - It's unclear how many bootstrapped demos this would even support, given their lengths in this use case. It may be possible to use a variant of BootstrapFewShot(), but it's still unclear if this program would be effective.
    - This could potentially raise the cost of the entire project by quite a lot due to the API pricing.
        - *QUESTION:* If the program is optimized using GPT-4o, does the same model need to be used for inference? It certainly seems plausible that

changing the model might render the optimization ineffective. If so, using GPT-4o in inference would make the cost skyrocket, as there are ~70000 legal case texts to process in the dataset.

- Break down the main program into smaller programs.
    - I have previously done this by separating a program with the same input, and whose only output is a 1-2 sentence summary of the doctor's violation(s).
    - Chunking the program by output fields wouldn't work at a larger scale because the input field is the problem. It might be possible to instead chunk the legal case text inputs themselves, and map each related piece back to the same parent.
        - For example, instead of processing 1000 token case_1 → output, this approach would process 500 token case_1a → output followed by 500 token case_1b → output. Then, the outputs for a case could be combined by performing bitwise OR across the output fields.
        - The optimization examples could also be tweaked to fit this structure. A concern is that the smaller input chunks will result in many of the inputs containing irrelevant information, thus producing all "no" or "not sure" for the corresponding output. It's unclear if the negative examples greatly outweighing the positive examples may create a bias toward negative output.
    - This would increase the cost, but not as much because the only redundant tokens are the output structure in the prompt, which happens because more queries are made.

Weighing the pros and cons, I'm leaning towards the last solution.