Robert Johns
CS7637: Knowledge-Based AI
Assignment 3
March 22, 2018

Commonsense from Nonsense: An AI Agent to Grade Mathematics

I. Introduction

I teach mathematics and computer science at a boarding school in Virginia. As such, I spend a good portion of my time grading papers. Too much. Teaching is great, but grading is awful, and if an AI agent were designed that could grade high-school mathematics accurately and with meaningful feedback, I would save hours every week and probably years of my life. Also, I could assign more homework.

Some attempts have been made in the field of auto-grading. For instance, most learning management systems include a quiz functionality, where answers can be graded based on how closely they match a pre-designated correct answer. These systems, however, are imperfect: they completely disregard the nuances of a multi-step process and frequently mark an answer as totally incorrect which was, in actuality, either off due to a small early mistake or correct and just poorly formatted.

On the other end of the spectrum, some services like WolframAlpha have implemented thorough step-by-step systems for solving any given problem involving algebra and calculus but have yet to leverage these systems into a grading scheme. An AI agent that could fairly assign points to a student's step-by-step solution process, while providing meaningful feedback along the way, could revolutionize mathematics education and would certainly sell.

II. Commonsense Reasoning & The Problem of Grading

Commonsense reasoning is said in our textbook to "[make] natural, obvious inferences about the world" (Goel, Joyner & Thaker, 2016, p. 200). At first, this may seem at odds with mathematics, which is based on deductive logic rather than real-world inferences, and while this is true when it comes to *solving* math problems, the question of *grading* math problems is completely different, and aligns quite nicely with commonsense reasoning. The ambiguity in inference, and thus, the extreme difficulty of the problem of grading, comes with reading an answer and interpreting what the student was thinking, and how much of the relevant material the student understands, from a few lines of written work.

One fundamental aspect of commonsense reasoning is that of *primitive actions*, a set of irreducible verbs that an actor can employ. For a human in a general situation, these are actions like "expel", "feel" and "conclude". For the problem of grading, we'll need to change our set of primitive actions. These can be actions like "divide both sides of the equation by 3", "take the integral of both sides with respect to *x*", or "multiply the numerator and denominator by the denominator's complex conjugate".

The second key aspect of commonsense reasoning is that of the *action frame*, an object containing a verb (primitive action) and other parameters that fit the situation. These action frames are put in sequential order to form a coherent story. In the same way, we could use action frames to represent each mathematical step a student may take in manipulating an equation or expression, and string those action frames together to form a solution to a given problem; the solution to a problem is analogous to a written story. These action frames can then be analyzed by the agent to identify work done correctly and mistakes made.

The final pillar of commonsense reasoning is the fundamental importance of background knowledge; for an agent to understand a written story, background knowledge (such as possible categories and semantic functions of verbs and prepositions) is key to making sense of a story. A grading agent, be they human or robotic, requires background knowledge in two forms. First, the agent must understand the mathematics being graded (otherwise anyone could do it and I'd be out of a job). Second, the agent must understand students and the kinds of mistakes they often make; building this set of knowledge is key to understanding errors and providing helpful feedback.

III. Agent Design

For simplicity and a place to start, we will say that the agent will be limited to grading problems based on solving equations and evaluating expressions (such as integrals) at all levels of mathematics up through single-variable calculus.

Given these parameters, let us set out to design our algorithm. A broad overview of the algorithm is below:

```
problem ←{problem statement, focus of problem, problem points}

agent_sols ← [set of agent-generated solutions to problem]

student_sol ← []
for each line of student's work:
        action_frame ← make_action_frame(previous line, current line)
        student_sol.append(action_frame)

state ←problem statement
current_points ←problem points

for each step in student_sol
        if not is_valid_approach(step, state):
                current_points -= determine_point_deduction(step, state, problem)
                terminate grading
        if not is_legal(step, state):
                current_points -= determine_point_deduction(step, state, problem)
                if the next step not in any solution in agent_sols:
                        terminate grading
        if step's input and output do not match its verb(s):
                current_points -= determine_point_deduction(step, state, problem)
        state = step
```

The process is clear, but some nuances deserve explanation. The "state" variable exists to define the current state of the problem, so that the next step's correctness can be determined; at the end of each step, the agent will update the "state" variable. The agent terminates grading if the approach of a step is completely invalid, as no further useful information can be gleaned if the solution careens off course. The agent will also terminate grading if an action is illegal (dividing by 0, for example) and the next step does not appear in any of the agent's generated solutions.

The agent's efficacy rests on five functions: the solution generator, is_valid_approach(), is_legal(), make_action_frame(), and determine_point_deduction(). The first function, while crucial, is beyond the scope of this paper; it's functionality will be much the same as WoflramAlpha's. The rest are equally essential and are heavily reliant on commonsense reasoning.

The function make_action_frame() will determine, based on two consecutive lines of student work, what primitive action or actions the student took in moving from one line to another. The result will be an action frame object as described below:

```
primitive_action [e.g. add to both sides]:

        input: [output of previous frame]

        quantity_used: [e.g. number added to both sides]

        output: [result of primitive action]

        subactions: [list of action frames
                containing small steps not
                explicitly stated]
```

To accomplish this task, perhaps this function could be built off of the solution generator: a sub-problem could be defined where the starting state is the frame's input and the ending state is the output; the primitive action is catalogued, and any small actions taken that are not explicitly stated are catalogued in the subactions field, as action frames of their own.

The functions is_legal() and is_valid_approach() have similar responsibilities, which are to determine if the fundamental idea behind the step a student takes is legitimate and valid (not the correctness of the details). These functions will use the step and the current state of the problem, along with mathematical background knowledge, to determine their return values. For example, if a student tries to add something to just one side of an equation, or take the arcsine of a value greater than 1, is_legal() will catch the error and call the function determine_point_deduction(), described below. If a precalculus student tried to solve a trigonometric equation by taking the natural logarithm of both sides, is_valid_approach() would spring into action.

The determine_point_deduction() function is responsible for determining the number of points to deduct for a given mistake. This deduction is determined by the number of points the problem is worth, the current state, and the step taken. Each of these values is crucial: if the problem is only worth five points, the agent should never deduct six, and a fundamental mistake coming early in a problem merits more of a deduction than a small mistake at the end. The difference between a fundamental mistake and a small mistake will be determined by the agent's background knowledge of mistakes; with this knowledge, the agent will be able to differentiate between a calculus student who has no idea what an integral is and one who simply forgot where the tangent curve's asymptotes are or who wrote that 2+2=5, as even the brightest among us do on occasion. This function will also be responsible for giving the student feedback on their work. The background knowledge used to determine how many points to deduct can also determine (or at least educatedly guess) what misconception the student had when they made their mistake and write a comment to the student describing and correcting their error.

The problem of structuring the background knowledge of mistakes is tricky, and depends on the idea of *goals*. In determining what steps constitute a valid approach, the agent will need to know the goal of the problem (or simply of the current state). For example, if the goal is to evaluate $\int x^2 dx$, adding 1 would be illegal and pointless. Perhaps the background knowledge could take the form of a production system: "if the goal is to solve an inequality and the current state is a polynomial < 0, find the zeroes and determine if the points between are positive or negative.

IV. Conclusion

While a grading agent is clearly difficult to implement, the power of commonsense reasoning makes the problem seem feasible, at least in the abstract. The potential usefulness of such an agent is beyond measure: in addition to removing the tedium of grading, the agent would have enough data to give students detailed statistical reports about what concepts are giving them trouble and need more work, an invaluable resource. If implemented well, such an agent has the potential to ease the lives of secondary school teachers' jobs worldwide, and improve mathematics education for us all.

Works Cited

Goel, Ashok; Joyner, David; Thaker, Bhavin (2016). *KBAI Ebook: Knowledge-Based Artificial Intelligence*. Publisher, city and state not given.