The City College of New York

CSC 59866-E: Senior Project I

AI Agents for Decision Making in the Real World

By Saptarashmi Bandyopadhyay
Email: sbandyopadhyay@ccny.cuny.edu, sbandyopadhyay@gc.cuny.edu
Assistant Professor of Computer Science
City College of New York and Graduate Center at the City University of New York

February 2, 2026 CSC 59866

How to do CS and AI Agents Research, How to Read Research Papers, & Introduction to Agentic AI Evaluation

How to do CS and AI Agents Research

Research Fundamentals

Computer Science research is not just about coding; it is the rigorous design and evaluation of a pipeline where:

1. An **input** is given
2. An **output** is received
3. The performance is measured using **metrics**

These three pillars define what your project does and how you can compare your project against the work of others.

The “Input”

Machine Learning models have many different kinds of inputs, from tabular data to text data, to image data and everything in between.

For AI Agents, the input often comes in the form of the *state (**s**)* of the agent’s environment or the agent’s *observation (**o**)* of part of the state (if the environment is not *fully observable*).

Assuming the agent knows everything (Oracle view) vs. what it actually perceives (Partial Observability) is often a cause of poor performance.

Examples (Input)

LLM Agent: Input is not just "the prompt," it is the "Context Window + System Instructions + Retrieved Documents (RAG)."

Robotic Agent: Input is not "the room," it is "LiDAR point clouds + RGB Camera frames."

Stock Agent: Input is "Time-series price history + News sentiment vectors."

The “Output”

For Machine Learning models, the output consists of a desired end result like a predicted value or generated image/text/audio.

For AI Agents, the output consists of an action (**a**) that the Agent takes, which has an impact on the Agent’s environment.

The action space (**A**)- the set of all possible actions the agent can take, is often what determines the complexity of the system as a whole

Examples (Output)

Discrete: Move Left/Right, Buy/Sell, Select Tool A/B. (Easier for RL).

- Example: Atari video game

Continuous: Joint velocity, Torque, Steering angle. (Harder, requires PPO/Soft Actor-Critic)

- Example: Autonomous Vehicle

The “Metrics”

The most common metric for Machine Learning model performance is accuracy, or how “correct” the final output was compared to a given answer.

"Accuracy" usually doesn't apply to Agents. Instead the agent receives a *reward (**r**)* from the environment at each step.

Better Questions:

- *Success Rate:* Did the agent achieve the goal? (Binary).
- *Efficiency:* How many steps did it take? What was the latency?
- *Safety:* Did it violate any constraints (e.g., hit a wall, hallucinate a fact)?

How to Read a Research Paper (Demonstration)

Proximal Policy Optimization (Schulman et. al. 2017)

This paper outlines ***Proximal Policy Optimization***, one of the most popular Reinforcement Learning algorithms used today.

It was created by OpenAI researchers in 2017, five years before OpenAI released ChatGPT.

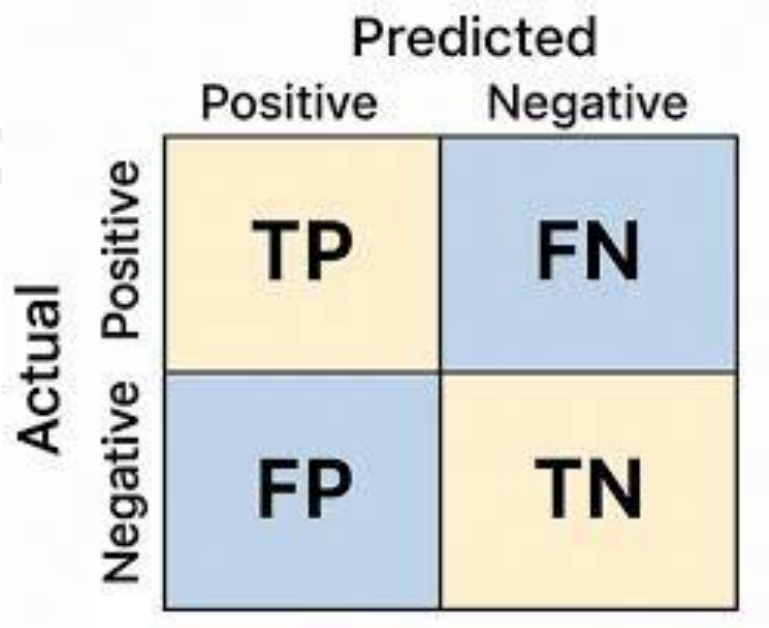
As we step through the paper, note what is *input*, what is *output*, and what are the *metrics*!

Proximal Policy Optimization Algorithms: https://arxiv.org/pdf/1707.06347

Introduction to AI Agent Evaluation

Confusion Matrix

- Visualizes performance of a classification model
- What kinds of errors does the model make? When is it the most accurate?
- Key Terms:
  - **True Positive**
    - Predicted True, Actual True
  - **False Positive (Type 1 Error)**
    - Predicted True, Actual False
  - **True Negative**
    - Predicted False, Actual False
  - **False Negative (Type 2 Error)**
    - Predicted False, Actual True
- Accuracy, Precision, Recall, and F-measure can be traced back to this!



Accuracy

- Accuracy Measures the percentage of predictions that are correct
- This is most useful when classes (positives vs negatives) are relatively balanced
- Warning: Can be misleading with imbalanced classes

$$\frac{TP + TN}{TP + TN + FP + FN}$$

Precision

- The proportion of positively identified examples
- Precision can be thought of as the percentage of the “caught” items that are actually in the positive class, as opposed to being mistaken

$$\frac{TP}{TP + FP}$$

Example: Precision in Distributed GPU Clusters

- When training an AI model across multiple GPUs in a cluster, different precision scores across different GPUs could signify an imbalance in your data distribution
- By tracking precision, you can make sure that the learning process is going smoothly across all GPUs
- If some GPUs have low precision, there may be problems with your model parallelization techniques

Recall

- The proportion of positives that were identified correctly out of all the positive examples
- Most important when the “cost” of a false negative is high, e.g. misdiagnosing a patient

$$\frac{TP}{TP + FN}$$

Example: Self-Driving Cars

- Because recall identifies what what ID'd correctly out of the total number of "hits", it's very useful for self driving cars
- Recall can track the total number of pedestrians and other hazards identified
- It can also be used to monitor important signals like stop signs, traffic lights, and speed limits
- Even slight mistakes could spell disaster!

F-measure

- The F-Measure is the harmonic mean between the precision and the recall
- It punishes an imbalance between the precision and the recall
- It's most useful when the number of positives and the number of negatives is lopsided

$$2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Why the harmonic mean?

- Imagine your model gets 99% of positive examples correct and 1% of negative examples correct
- If you were to take the arithmetic mean (aka the accuracy), you would get 50%
- This doesn't consider at all the failure to correctly identify all examples
- This same model would have an F-measure of ~2%, clearly letting you know the model needs debugging

Example: Monitoring Autonomous Systems

- F-measure can be used when both precision *and* recall are important
- If you're operating a computing cluster, it's important to both:
  - Identify malfunctioning nodes accurately (high recall)
  - Correctly flag faulty nodes (high precision)
- More applications:
  - Intelligent job prioritization (e.g. SLURM clusters)
  - Autonomous Management Systems

When do I use each metric?

- Start with the **confusion matrix**
  - This will give you solid footing for exploring the other metrics and understanding what kinds of errors your model is making
- Use **accuracy** when you want a quick glance at how the errors across your (balanced) dataset are distributed
- Choose **precision** when avoiding False Positives is important, and **recall** when False Negatives are important
- **F-measure** is most useful as a reliable metric for unbalanced positives vs negatives in your dataset

Questions?