

CS224V Homework 2

Part 1: Building ReAct Agents and adding them to AOS Part 2: Formal Trial Matching with SatIR and VERDICT

Stanford University

Due: 2026-10-18 23:59 PDT

1 Part 1: Building Simple Agents

In the previous assignment, you were given a **rare disease** and a cohort of patients. Their symptoms and clinical findings were encoded as Human Phenotype Ontology (HPO) terms, standardized labels for findings such as seizures. Out of 10 patients, exactly one patient had the target disease. The task was to identify that patient by comparing their findings with evidence about the disease.

In this assignment you will learn how to build agents, using a method called ReAct and then we will learn how to use Agentic OS (AOS) for orchestrating agents.

ReAct combines reasoning and acting. The model gathers information, uses what it learns to choose its next step, and repeats until it can answer. This approach comes from ReAct (Yao et al., 2023)

AOS is an open, accountable platform for agents with a layered architecture that includes a semantic file system for retrieving and curating persistent knowledge and artifacts; a harness with experimental components to improve reasoning, context management, conversation management, and decision making; team workflow support for shared workspaces and collaboration. Together, these components provide a common substrate for building and coordinating advanced agent systems. AOS can provide a good foundation for your projects!

You will use the notebook to complete the homework. All the information you need to complete the assignment is already included.

[Link to Google Colab Notebook](#)

Use the same disease assignment from Assignment 1: [Disease Assignments](#)

Before you begin, please make a copy of the notebook and remember to save your work regularly.

After you run the final cell, the notebook will generate a `answers.json` file. Please submit this file along with your completed notebook.

Grading. There are 22 TODOs worth 110 marks in total (Table 1): 16 TODOs worth 57 marks in Part 1, and 6 TODOs worth 53 marks in Part 2. The Part 2 TODOs carry the prefix P2- and are described in Section 3.

Table 1: Assignment tasks and marks

TODO	Task	Marks
1	Make your first LLM call	2
1.2	What tools would you use?	4
2	Agent Prompt	4
3	Create langchain agent	2
4	Provide user message	2
5	Retrieve conversation results	2
5.1	Tool call questions	4
6a	Structured output A	2
6b	Structured output B	2
6c	Structured output C	2
6d	Structured output D	2
6e	Structured output E	2
7	Initialize AOS	4
8	Step through AOS SLIDERS	4
9	Putting your ReAct agent into AOS	4
10	Compare using a new agent	15
P2-1(a)	Which step is the fragile one?	6
P2-1(b)	Checking what the policy filled in	8
P2-2(a)	Limits of the gate	8
P2-3(a)	Coordinator versus regulator	8
P2-3(b)	Fixed policy versus model judgment	8
P2-4	Combining GRILL with SatIR/VERDICT	15

2 Sign up for Google Credits

We are proud to announce that we have Google Credits for students to work on their assignments and projects! We will send the mail to enrolled students!

3 Part 2: Turning Clinical Text into Constraints

To decide whether a patient qualifies for a clinical trial, someone has to compare two documents written in prose: the patient’s chart and the trial’s eligibility criteria. A common approach today is to give both documents to a language model and ask it for an answer.

Part 2 studies a different approach. A language model first **translates** the criteria and the chart into SMT (satisfiability modulo theories): named variables and logical constraints. A solver then **decides** eligibility from those constraints. You will see what this makes possible, namely searching thousands of trials quickly and checking *why* a decision was made, and where it falls short.

The central goal of Part 2 is to understand **what the language model does, what ordinary**

code does, and where the hand-off between them can fail. The code treats the model's translation as correct, so if the translation is wrong, the code will be precisely wrong. You should be able to explain how a decision was reached, what it rests on, and what would change it.

Most of the infrastructure is provided. Every model step and every solver run was done in advance, and the real outputs are saved inside the notebook. Unlike Part 1, there is no API key to set up, no model call to make, and nothing to install. Part 2 asks you to read each stage, inspect what it produced, and explain what you see.

Part 2 has its own notebook. All the information you need to complete it is already included.

[Link to Google Colab Notebook for Part 2](#)

Before you begin, please make a copy of the notebook and remember to save your work regularly.

After you run the final cell, the notebook will generate a `todo_answers.json` file. Please submit this file along with your completed Part 2 notebook. In total you submit four files: `answers.json` and your completed Part 1 notebook, and `todo_answers.json` and your completed Part 2 notebook.

Part 2 Structure: The Part 2 notebook has four parts of its own, following the pipeline from text to decision. To avoid confusion with Homework 2's Part 1 and Part 2, we call them Notebook Part 1 to Notebook Part 4 here.

- **Notebook Part 1: Text becomes constraints.** Follow one real patient and one real trial from prose to variables and assertions, and read the solver's verdict.
- **Notebook Part 2: The same constraints, used for search.** Turn constraints into database rows and see how they are used to search thousands of trials.
- **Notebook Part 3: Auditing a decision.** Examine one decision through four questions: policy, assumptions, pivotal conditions, and self-faithfulness.
- **Notebook Part 4: Combining the two.** Argue how this approach could improve GRILL from Homework 1 on rare-disease diagnosis.

Grading for Part 2. The Part 2 rows of Table 1 (P2-1(a) to P2-4) reward reasoning, not agreement with us: an answer that argues a different position clearly and with evidence from the notebook scores higher than one that repeats the notebook without explaining it. Every claim about what the notebook shows must point to the output it relies on, for example by quoting the pair id, the requirement name, or the heading of the cell.