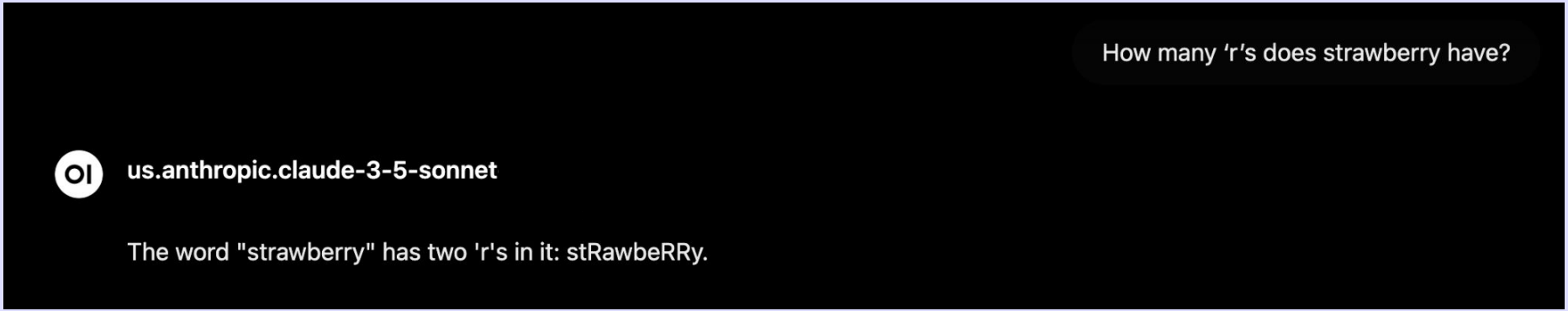
So you think building tools for agents is easy?

01

What are tools?

LLMs have limited capabilities!



Tools are ways for LLMs to perform actions

Input Schema

```
{
    "type": "object",
    "name": "Character Counter Tool",
    "inputs" : {
        "character": {
            "type": "string",
            "description": "The character to search for and count"
        },
        "sentence": {
            "type": "string",
            "description": "The text in which to count character
              occurrences"
        }
    }
}
```

Tool Description

The character-count tool lets you count the number of occurrences of a particular character (specified by 'character' parameter) in a particular string (specified by 'sentence' parameter)

Tool Implementation

```
def count_character(character: str, sentence: str) -> int:
    return sentence.count(character)
```

That seems really trivial. What am I missing?

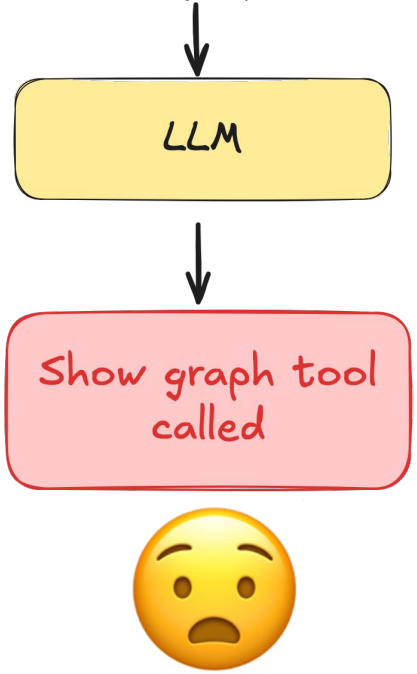
```
{
    "name": "Visualization Creator",
    "description": "Creates a visualization based on the specified type and styling",
    "parameters": {
        "chart_type": {
            "type": "string",
            "enum": ["bar", "line", "scatter", "pie", "heatmap"],
            "description": "Type of visualization to generate"
        },
        "line" : {
          "x" : {
              "type" : "array",
              "items" : {"type" : "number"}
          },
          "y" : {
              "type" : "array",
              "items" : {"type" : "number"}
          }
        },
        .
        .
        .
        "styling": {
          "color_scheme": {
              "type": "array",
              "items": {"type": "string"},
              "description": "Array of hex color codes"
          },
          "font_settings": {
            "family": "string",
            "size": "number",
            "weight": "string"
          },
          "dimensions": {
            "width": "number",
            "height": "number"
          }
        }
    }
}
```

This tool creates data visualizations based on structured input data. It accepts a data source (containing values, labels, and metadata), chart type selection (bar, line, scatter, pie, or heatmap), and optional styling and interactivity parameters. Choose bar charts for categorical comparisons, line charts for temporal trends, scatter plots for variable relationships, pie charts for proportional data, and heatmaps for matrix-style data patterns. The tool handles visualization formatting, applies specified styling (colors, fonts, dimensions), and implements interactive features like hover effects and animations. To use, provide the required data structure and specify your desired chart type - the tool will generate an appropriate visualization following data visualization best practices.

What starts to go wrong?

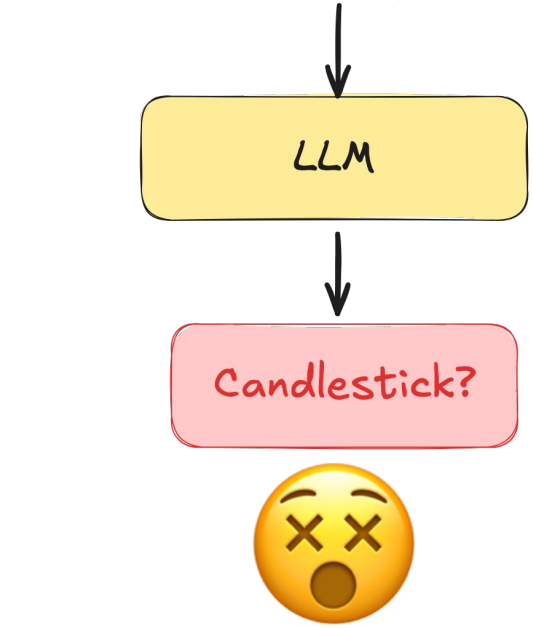
CLASSIFICATION FAILURE

Show me the data as a graph...



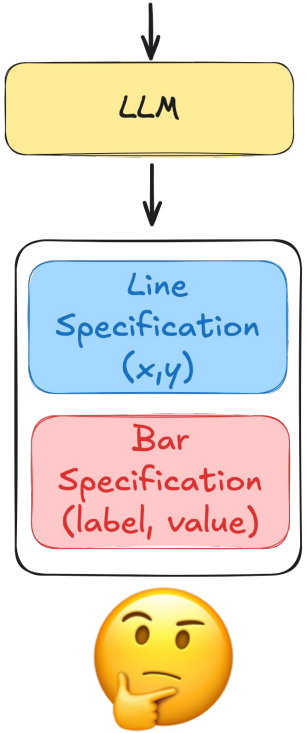
SCHEMA NOT FOLLOWED

Allowed Values :
line, bar, scatter, pie, heatmap



TOOL-USE CONFUSION

Give me the visualization as a line OR a bar...

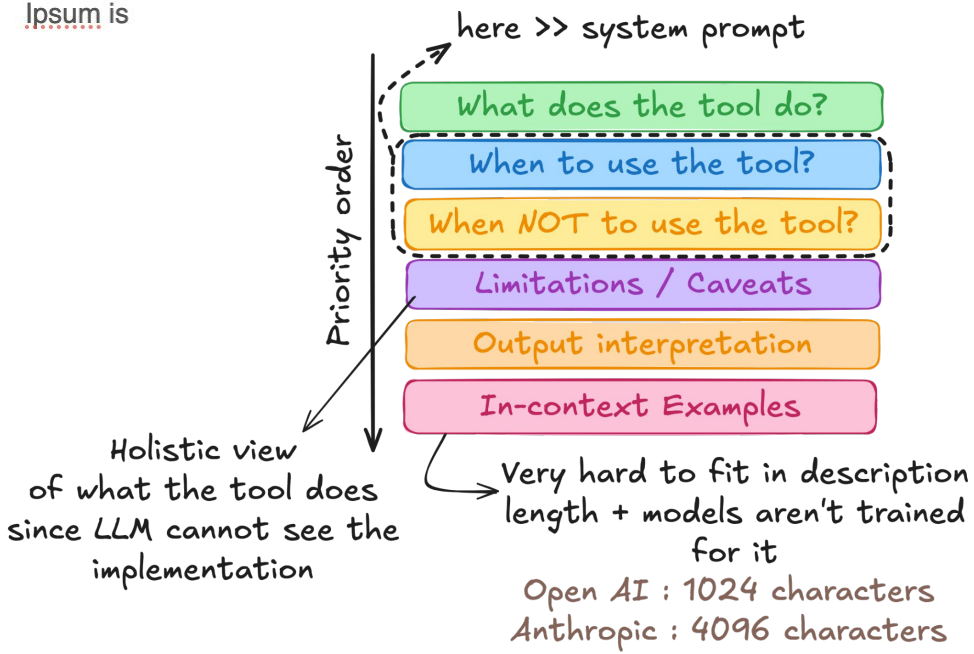


02

How do we mitigate these issues?

Structuring your tool-description

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum. Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets Lorem Ipsum is

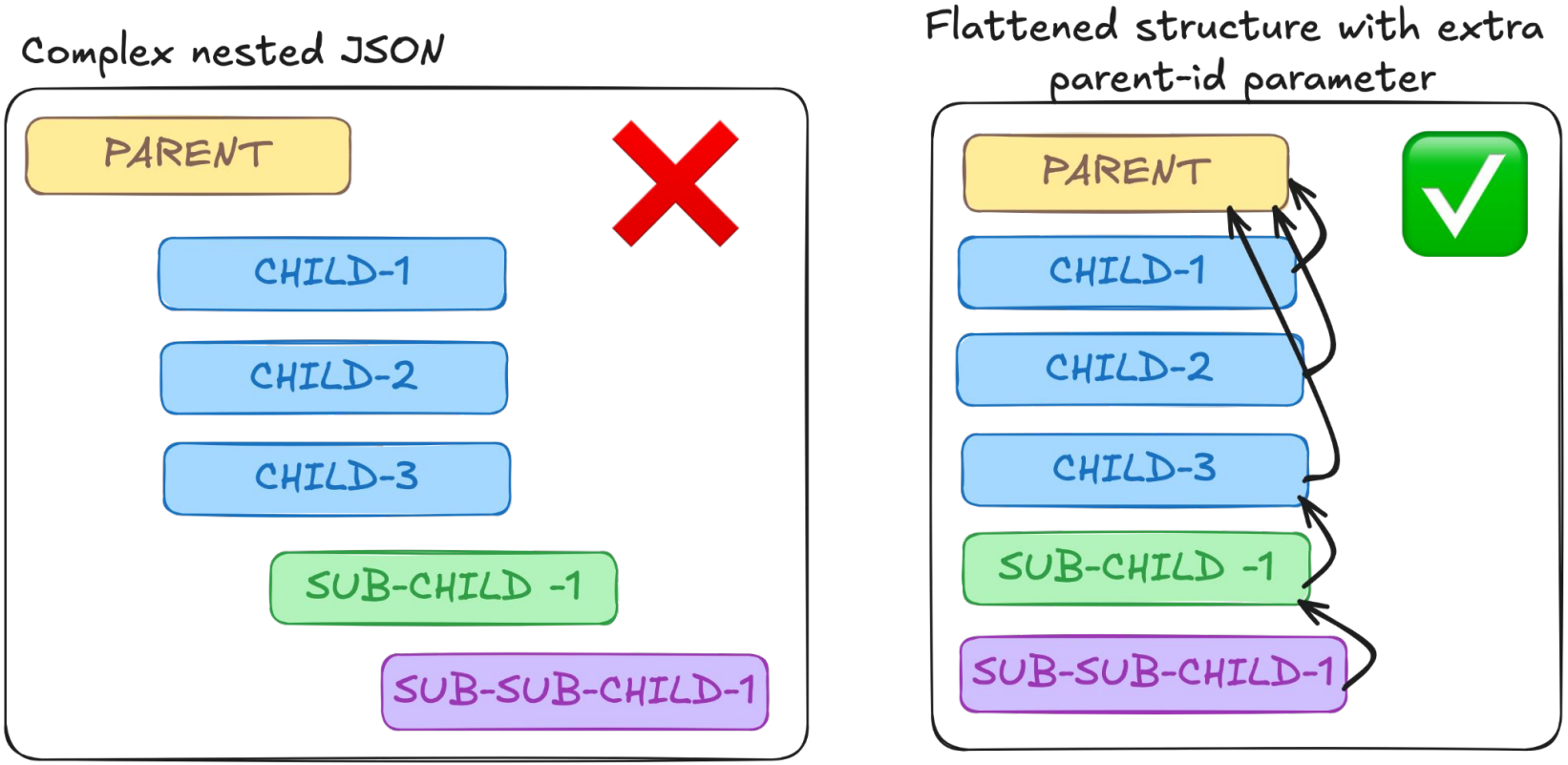


Put this in the system prompt if absolutely required

- How to respond to the user after using the tool
- Input parameter specific examples / explanations
- What tool should be used before or after
- Complex in-context examples

These changes gave us ~30% tool-use classification accuracy improvements!

Flatten tool schemas as much as possible



Don’t use words. Use Structure

Tool-Schema :

```
{
    json_value : List[int, string, null]
}
```

Tool description :

Use only strings when specifying IDs
Use ints when specifying enum values
For absolute dates :
- After : [date, null]
- Before : [null, date]
-Between : [date-1, date-2]
To check if its empty, specify a an empty list (e.g. [])

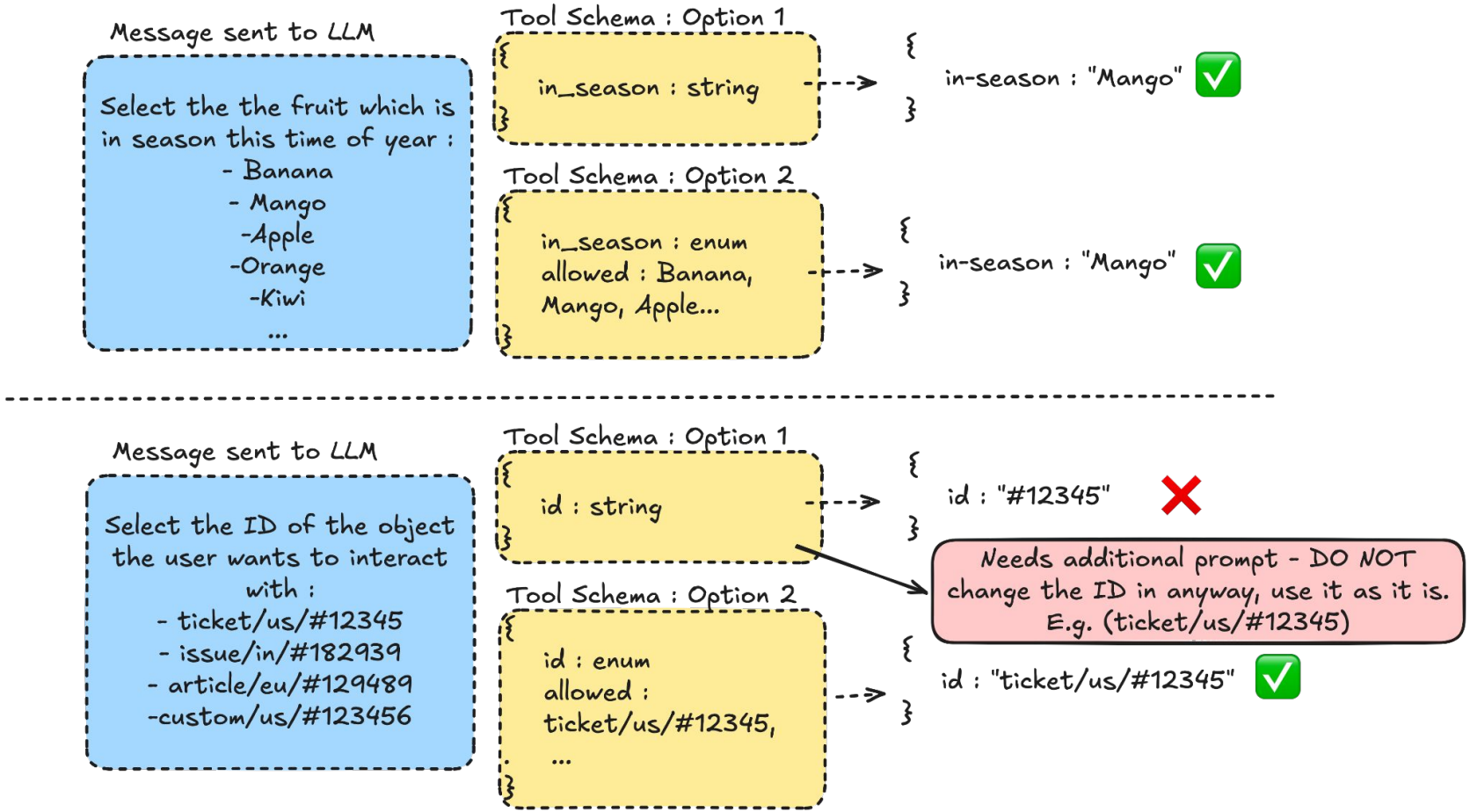
Tool-Schema :

```
{
    any_of : [
        id_specification : List[string],
        enum_specification : List[int],
        absolute_date_spec : {
            before : string,
            after : string
        },
        is_empty_specification : boolean
    ]
}
```

Tool description :

This small change alone, gave us ~20% tool-use accuracy jump, entirely zero-shot

Constrain LLM output where possible



03

How do you systematically improve performance?

Benchmarking!

Tool-specific metrics
E.g : right values for the parameter?

Independent Tool-Use

How? Force the LLM to use your tool on every request
Why? Determine how clear your tool-schema is

Standard classification metrics: accuracy, recall

Independent Tool-Use Classification

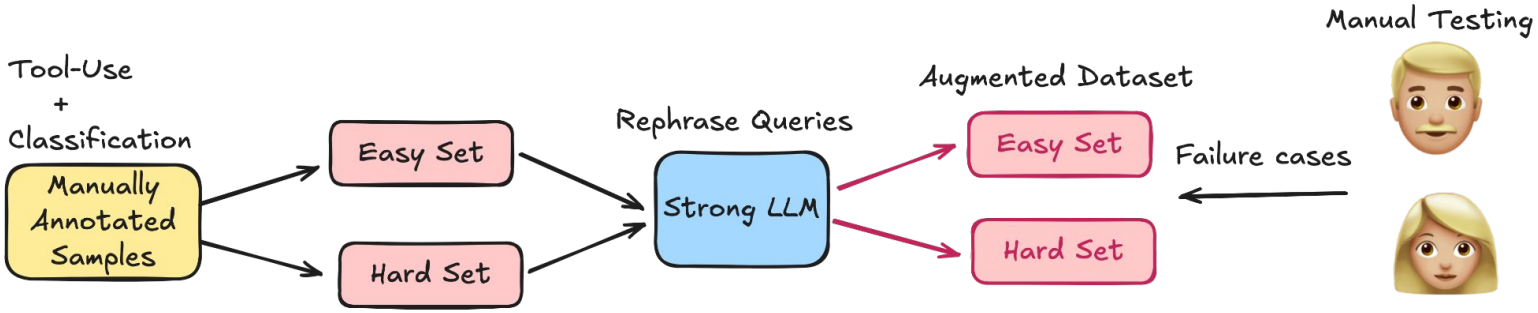
How? Make the use of the tool optional instead of forced
Why? Determine how clear your tool-description is

Tool-specific + Classification metrics

Tool-Use & Classification with many other tools

How? Add other tools & make tool use optional
Why? Determine how well your tool works in an actual agent

Dataset Creation Pipeline Example



Other questions : How much data is enough? How many times should you re-run the benchmark?

04

Benchmark score : 100%
Deployed agent status : FAILING?

How do you debug an agent?

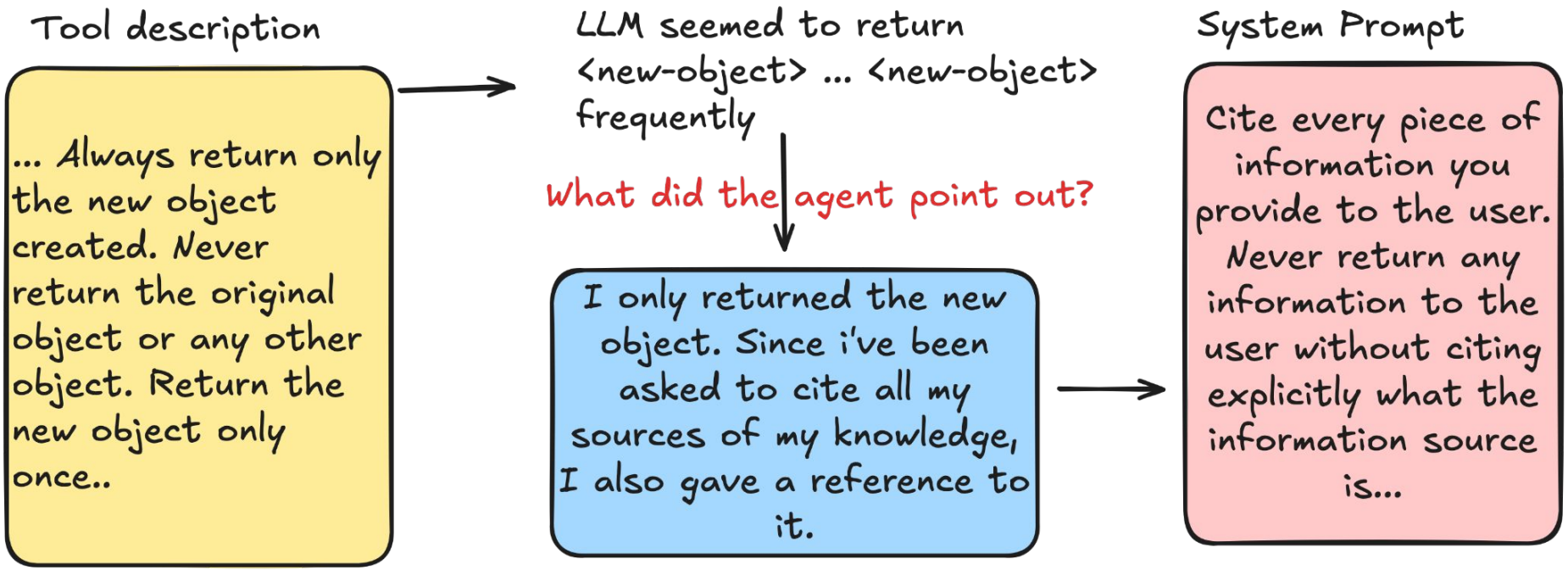
1. Try to reproduce the failure (not always possible)

2. Ensure that the prompt, the schema, and extra information correctly got to the LLM

3. If the above two fail : ASK THE AGENT WHAT MISLEAD IT!

X was what you were supposed to do, Y is what you did. Pinpoint exactly what parts of the prompt, the tool description or schema mislead you or are conflicting.

Don’t think this works?



05

What's a good sequence of steps to build effective tools?

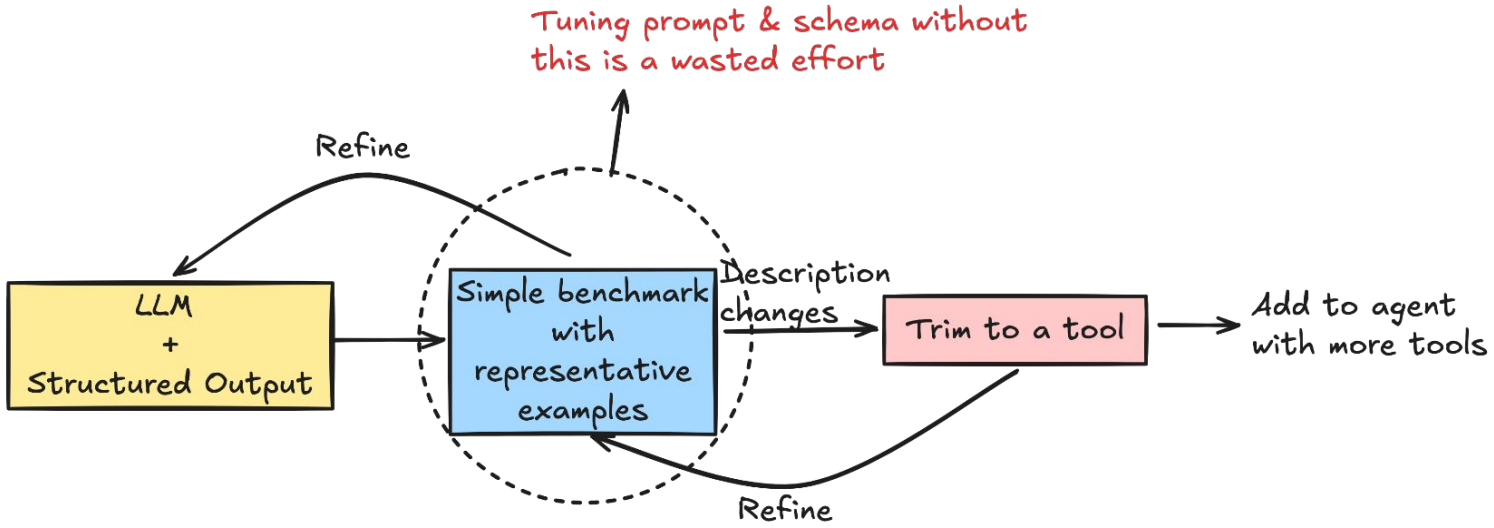
Development Flow

- Not constrained by description length.

- You can figure out a good tool-schema with free iteration

- Gives a reasonable upper bound from the get go.

- Lets you fall back on Agent as a tool, if a simple tool doesn't work



THANK YOU!