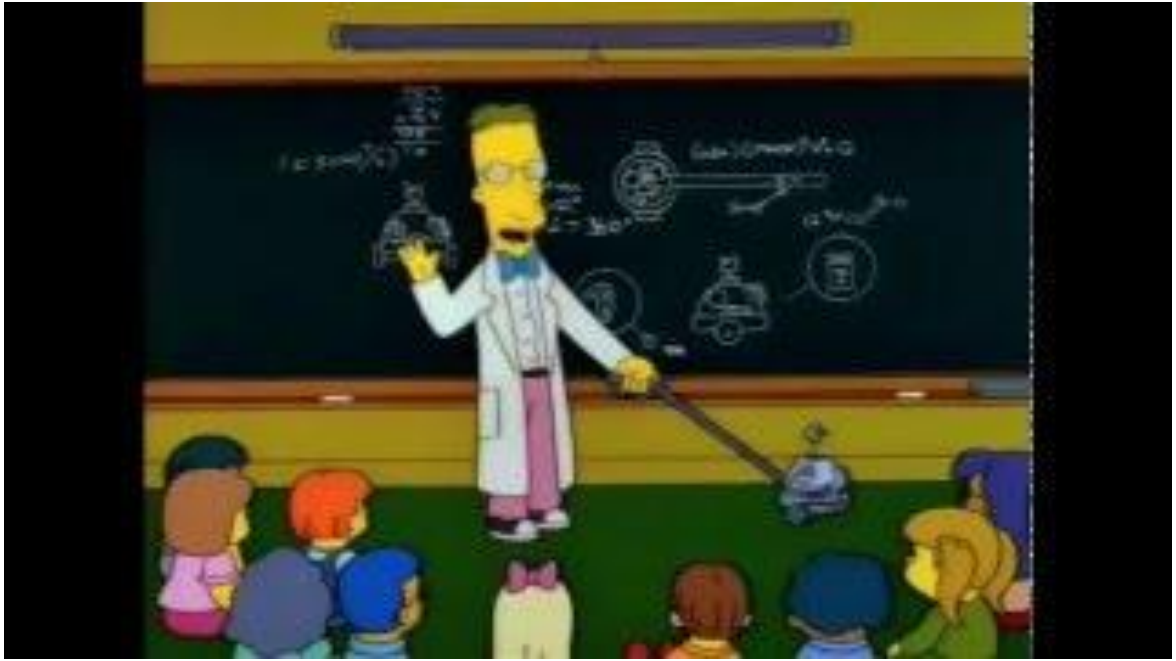
Documentation is Code (with agentic AI)

Allow billions of people to build billions of solutions

You won’t enjoy it on as many levels



What is DisC

- Documentation is Code(DisC)
- Any solution where instructions written using natural language are *executed* by an Agent

Working with Agentic AI

- What is an agent?
- High learning curve
- LLMs produce excessive output
- Inability to modify output
- No empathy, no mercy

Food for thought

- What if we are not looking for an answer?
- What if it is ok to make mistakes?
- Maybe Agents are not supposed to solve every problem?

Evolution of Code

- Tubes, punch cards, assembly tapes
- Higher level languages to meta-frameworks
- No code
  - a GUI that builds things for you
  - Millions of products
- Documentation is Code
  - Just write the requirements and execute

Billions of people to build billions of solutions

- What about runtime performance?
- f(problem) = solution
  - The solution is a function of the problem
- Benefits outweigh the costs
  - Proof of Concept
  - Personal projects
  - Internal tooling
- Multi-modal zero code solutions

Anatomy of DisC

- End user will only use natural language
- Minimize software engineering concepts
  - Unstructured data
  - Implicit algorithms
- Dynamic tools - no code changes for adding tools
- Human in the loop built-in
- Consistent output
- Iterative development

Demo

- Build a dashboard for our github repo
  - The requirements keep changing
- Start with a traditional solution
- Then transform to a DisC solution

Requirements

- For the most prolific author of github issues, give me the number of issues they have created.

- Given the author's issues, I want to know the number of comments that same author has made.

Frontend

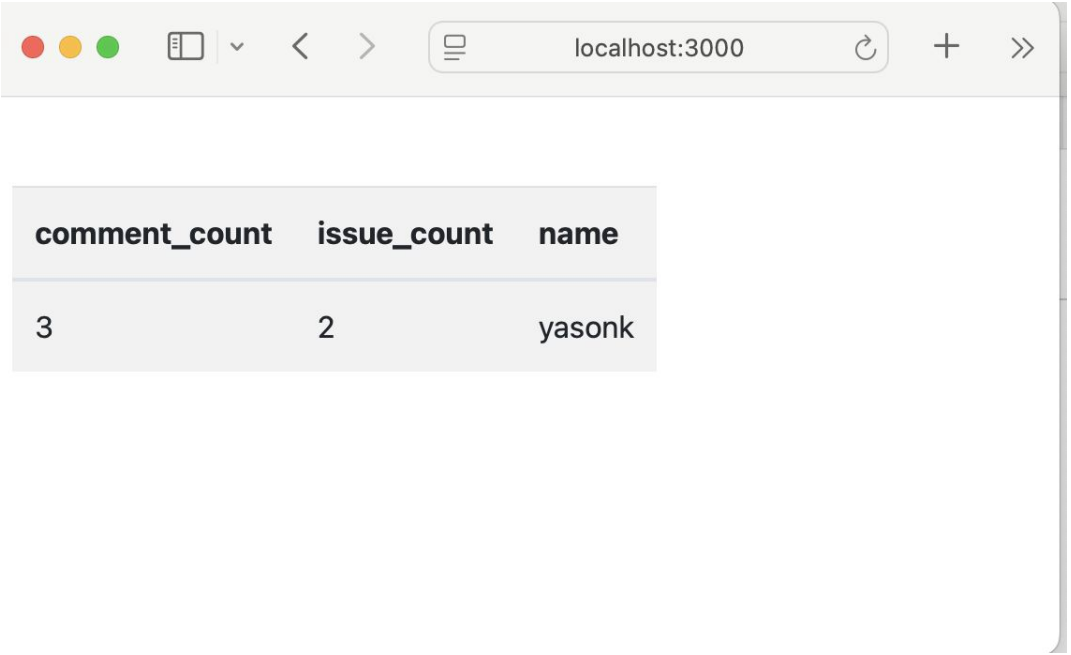
```
...

<div class="container mt-5">
        <table class="table table-striped">
            <tbody id="dataTable">
                <!-- Data will be populated here -->
            </tbody>
        </table>
    </div>

    <script>
        async function loadData() {
            const response = await fetch('http://localhost:5001/data');

...
```

comment_count	issue_count	name
3	2	yasonk

Backend

```
for issue in issues:
  author = issue["user"]["login"]
  author_count[author] = author_count.get(author,0) + 1

author = max(author_count, key=author_count.get, default=None)

issue_count = author_count.get(author, 0)

comments_count = sum(1 for comment in comments if comment["user"]["login"] == author)


result = {
    "data": [
        {"name": author, "issue_count": issue_count, "comment_count": comment_count}
    ]
}
return jsonify(result)
```

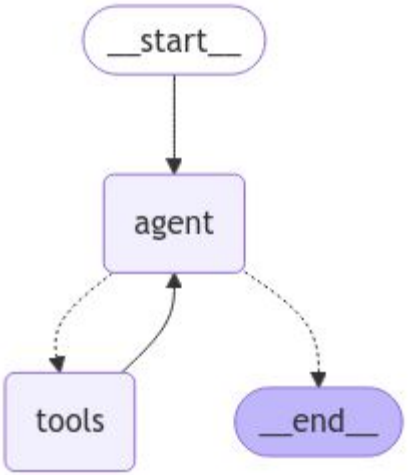
Result



comment_count	issue_count	name
3	2	yasonk

ReAct Agent

- Reason - Act - Observe - Iterate loop
- ReAct : Synergizing Reasoning And Acting In Language Models https://arxiv.org/pdf/2210.03629
- Tool calling support
- Prebuilt LangGraph ReAct agent



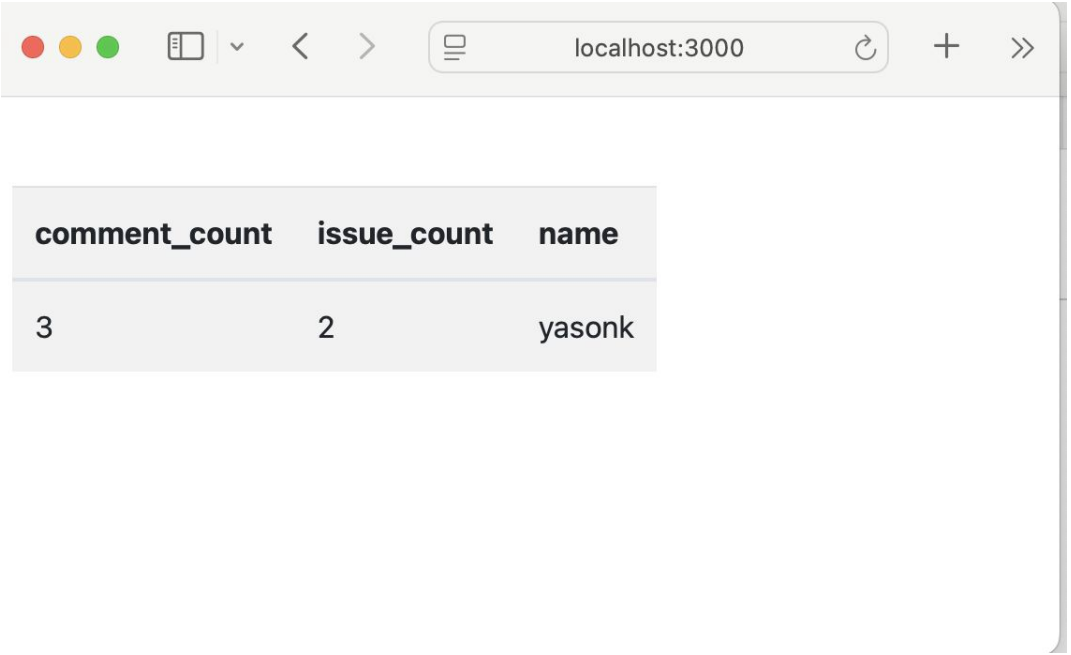
ReAct Agent Implementation

```
# Precoded tools (not dynamic)
toolkit = GitHubToolkit.from_github_api_wrapper(GitHubAPIWrapper())
tools = [setattr(tool, "name", tool.mode) or tool for tool in toolkit.get_tools()]

llm = ChatOpenAI(model="gpt-4o", temperature=0)

def ai(query: str, response_format: Type[BaseModel]):
    agent_executor = create_react_agent(llm, tools,response_format=response_format)
    events = agent_executor.stream({"messages": [("user", query)]}, stream_mode="values")

    event = {}
    for event in events:
        event["messages"][-1].pretty_print()

    structured_response = event.get("structured_response")
    return json.dumps({"data": [structured_response.model_dump()]}
```

Route Implementation

```python
class Author(BaseModel):
    name: str
    issue_count: int
    comment_count: int


@app.route('/data', methods=['GET'])
def get_data():
    query = ("""
For the most prolific author of github issues, give me the number of issues they have created.
Given the author's issues, I want to know the number of comments that same author has made.
""")

    return ai(query, response_format=Author)
```

Result



comment_count	issue_count	name
3	2	yasonk

Route Implementation

```python
class Author(BaseModel):
    name: str
    issue_count: int
    comment_count: int


@app.route('/data', methods=['GET'])
def get_data():
    query = ("""
For the most prolific author of github issues, give me the number of issues they have created.
Given the author's issues, I want to know the number of comments that same author has made.
""")

    return ai(query, response_format=Author)
```

~~Dynamic Tools~~

~~Unstructured Data~~

Implicit Algorithms

~~Uses Natural Language Only~~

New Requirements

- Find the most prolific author of github issues, and I want to see all of their issues and comments
- I will need to see only the author, issue title, and comments.

Open Jupyter Notebook

Capabilities

- What did we achieve in 30 minutes?
  - End user will write only natural language
  - Unstructured data
  - Implicit algorithms

Limitations

- Tools still need to be coded
- Must learn agent specializations and limitations
- Runtime performance is slow
  - But should always be 100x faster to develop
- Inconsistent outputs
  - Blasts out overreaching solutions
  - Incremental refinement is challenging
- Engineering rigor
  - Security
  - Scalability
  - Debugging

Next Steps

- Tools to be dynamic without requiring new code
  - Simply drop in documentation for a tool
- Stateful agents
  - Checkpointing and persistence
  - Iterative development
  - Output consistency
- Improve human-in-the-loop processes
  - Human in the loop is unavoidable
  - Assist with debugging

Are we obsolete now?

- Expert knowledge is still needed
  - Expertise on building Agents
  - Communicating with AI agents
  - Retooling for AI (terminals, browsers)
  - Re-documenting for AI
- Need better LLMs

Conclusion

- With DisC, users will write their specification and the agent will run it
- Solve of problems rarely solved before
  - Developer setup instructions
  - A small website for your party
  - A tennis league at work
  - A behavior tracking app for you child
- Billions of people will be able to solve billions of highly personalized problems

Link to this presentation

My Profile