* Building Agentic RAG
  ↳ a framework to help you build Research agents.

* Autonomous Research Agent.

Building a full agent:

Step 1: Routing

We add decision making to route requests to multiple tools.

Step 2: Tool use.

where you create an Interface (UI) for agents to select a tool as well as generate right arguments for that tool.

Step 3: Multi step reasoning with tool use.

We use an LLM to perform multistep reasoning with a range of tools for retaining memory through out that process.

Create a higher level Research Assistant.

Ensure Debugging on LLM, user optionally enter guidance in middle steps.

* Building Router Over a single document that could handle both question and Answering and summarization.

* Router Engine.

Simple Directory Reader to read

Routers → How to use LLM's to make a decision by picking a choice of different pipelines.

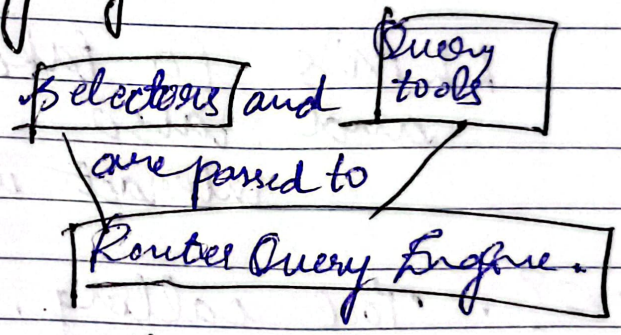
* Selectors to build Router.
  ↓
  has distinct attribute.

LLM selector: prompting an LLM to output a JSON that is parsed, and the corresponding indexes were queried.

Pydantic selector:

use the openai function calling API to produce pydantic selection objects rather than parsing raw JSON.

Defining Router Query Engines



This will select the right tool as per prompt.

* Tool Calling.

Not only pick a function to execute but inference arguments to pass through the function.

* Tool calling continuations-

One of the promises of our LLM's is their ability to take action and interact with External Environment.

But how do we use LLM's to take action and interact with external Environment?

* LLM's to take action and also interact with External environment?

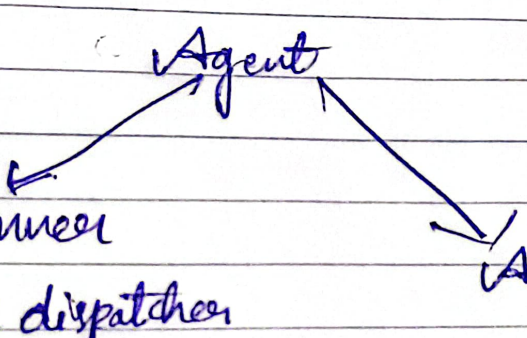
* How? . Tool calling .

Tool calling helps LLM's to interact with external environment through a dynamic interface where tool calling helps :-

:) Choosing the right tool & inference right argument. (FunctionTool)
Predict and call function not only helps picking tool but also helps in setting parameters.

* Building an Agent Reasoning Tool.
* Agent reasoning loop over tools to pick the right one for multiple steps.
* Function calling

Agent in LlamaIndex.



- Overall task dispatcher
- Agent task Orchestrator.
- Agent states mapping from task id to taskstate.

Task states
- Task
- Completed steps.
- step Queue.

Memory:
Conversation Memory.

- Task reasoning and Execution:

Tools:
Vector Tool
summary Tool.

LLM: LLM to use.

shared message tool & subscription mechanism.