

## RAG Meets AI Agents: Unlocking Agentic RAG with PydanticAI!



YourTechBud Codes  
5.48K subscribers

Subscribe



77



0



Share



Ask



Download

...

2,558 views Apr 26, 2025 #AI #GenerativeAI #GenAI

We're combining the magic of Retrieval-Augmented Generation (RAG) with the adaptability of agent-based AI, powered by PydanticAI. Whether you're an AI enthusiast or a pro developer, this session is packed with mind-blowing insights and real-world value! 🎉

Here's the Agenda:

- 1] Spot the Limitations of RAG: Not every problem can be solved with standard RAG pipelines. Learn how to pinpoint those challenging use cases where a little "agentic thinking" can save the day.
- 2] Discover Agentic RAG Awesomeness: Find out how combining RAG with agentic principles opens up a world of possibilities. Trust us, it's a game-changer!
- 3] Real-World Action: Dive into a live use case and watch Agentic RAG in action as we push its boundaries in a practical scenario.

...

References:

- 1] PydanticAI Documentation: <https://ai.pydantic.dev/>
- 2] Code: <https://github.com/YourTechBud/ytb-pr...>

Chapters:

- 00:00 - Introduction
- 00:17- A RAG Recap
- 03:44 - Where naive RAG falls short?
- 06:01 - Improving our RAG pipeline
- 11:12 - Why use agents for RAG?
- 13:55 - Using PydanticAI for RAG?
- 31:03 - Building our agents
- 35:39 - The final agentic RAG pipeline
- 39:13 - Seeing it in action
- 41:47 - Making it even harder

Name	Last commit message	Last commit date
..		8 months ago
.vscode	Added code for agentic rag with pydantic	8 months ago
data	Added code for agentic rag with pydantic	8 months ago
src	Added code for agentic rag with pydantic	8 months ago
.env.example	Added code for agentic rag with pydantic	8 months ago
.gitignore	Added code for agentic rag with pydantic	8 months ago
.python-version	Added code for agentic rag with pydantic	8 months ago
README.md	Added code for agentic rag with pydantic	8 months ago
pyproject.toml	Added code for agentic rag with pydantic	8 months ago
uv.lock	Added code for agentic rag with pydantic	8 months ago

## Agentic RAG with Pydantic AI

Consider giving this repo a ⭐! Thanks!!!

### Prerequisites

You need to have the following tools installed:

- [uv](#)
- [Inferix](#) or any OpenAI compatible API.

### Environment Setup

#### 1. Setup uv

```
# To setup uv
uv sync
```

agentic-rag-pydanticai
> .vscode
> data
> src
.env.example
.gitignore
.python-version
README.md
pyproject.toml
uv.lock
> autogen-agent-workflow-patterns
> autogen-kbs-basic
> autogen-workflows
> baml-agents
> building-agents-with-mcp
> crawlai-101
> generative-ai-roadmap-2025
> get-started-with-istio-3-steps
> google-cloud-functions-introduct...
> google-cloud-run-introduction

## 2. Install Inferix (OpenAI compatible API)

Note: You can use any OpenAI compatible backend or simply use OpenAI itself.

- Feel free to use any OpenAI compatible API.
- Make sure you have [Ollama](#) installed.
- Make sure you have pulled a model. I recommend [Qwen 2.5 32B](#).
- Use this [guide to setup inferix](#) to host a OpenAI compatible API capable of function calling.

## 3. Set the environment variables

- Checkout the example dotenv file at `.env.example`.
- Create a new `.env` using the example one as a template.
- Replace the variables as necessary.

## Run the app

We've got a few scripts in this project:

1. `uv run src/00_humble_call.py` - Runs the ingestion script to create the embeddings and full text search index.
2. `uv run src/01-basic-rag.py` - Demonstrates a basic RAG workflow.
3. `uv run src/02a-rag-as-tool.py` - Demonstrates how you can agents can perform RAG by means of a tool call.
4. `uv run src/02b-rag-as-system-prompt.py` - Demonstrates how you can use Pydantic AI's system prompt hook to perform RAG.
5. `uv run src/02c-rag-combined.py` - Demonstrates how you can combine the two approaches to perform RAG.
6. `uv run src/03-agentic-rag.py` - Demonstrates how you can use Pydantic AI's agentic workflow to perform RAG.

https://github.com/YourTechBud/ytb-practical-guide/tree/master/agentic-rag-pydanticai/data

Name	Last commit message	Last commit date
..		
charizard.md	Added code for agentic rag with pydantic	8 months ago
mewtwo.md	Added code for agentic rag with pydantic	8 months ago
pichu.md	Added code for agentic rag with pydantic	8 months ago
pikachu.md	Added code for agentic rag with pydantic	8 months ago
raichu.md	Added code for agentic rag with pydantic	8 months ago
slowpoke.md	Added code for agentic rag with pydantic	8 months ago

https://github.com/YourTechBud/ytb-practical-guide

Name	Last commit message	Last commit date
..		
agents	Added code for agentic rag with pydantic	8 months ago
00-ingestion.py	Added code for agentic rag with pydantic	8 months ago
01-basic-rag.py	Added code for agentic rag with pydantic	8 months ago
02a-rag-as-tool.py	Added code for agentic rag with pydantic	8 months ago
02b-rag-as-system-prompt.py	Added code for agentic rag with pydantic	8 months ago
02c-rag-combined.py	Added code for agentic rag with pydantic	8 months ago
03-agentic-rag.py	Added code for agentic rag with pydantic	8 months ago
utils.py	Added code for agentic rag with pydantic	8 months ago

https://github.com/YourTechBud/ytb-practical-guide/tree/master/agentic-rag-pydanticai/src/agents

YourTechBud / ytb-practical-guide

Code Issues Pull requests Actions Projects Security Insights

Files master + Go to file

Istio Practical - Traffic Control - P... adk-multi-agent agentic-rag-pydanticai .vscode settings.json data charizard.md mewtwo.md pichu.md pikachu.md raichu.md slowpoke.md src agents extractor.py

ytb-practical-guide / agentic-rag-pydanticai / src / agents /

YourTechBud Added code for agentic rag with pydantic 2afdb5d · 8 months ago History

Name	Last commit message	Last commit date
..		
extractor.py	Added code for agentic rag with pydantic	8 months ago
finalizer.py	Added code for agentic rag with pydantic	8 months ago
planner.py	Added code for agentic rag with pydantic	8 months ago
retriever.py	Added code for agentic rag with pydantic	8 months ago

Pikachu generation 3 move learnset & egg move parents (Ruby, Sapphire, FireRed, LeafGreen, Emerald) | Pokémon Database

Pokémon data Game mechanics Pokémons Community/Other Search

## Pikachu - Generation 3 learnset

This page lists all the moves that Pikachu can learn in Generation 3, which consists of these games:

- Pokémon Ruby
- Pokémon Sapphire
- Pokémon FireRed
- Pokémon LeafGreen
- Pokémon Emerald

At the bottom we also list further details for egg moves: which compatible Pokémons can pass down the moves, and how those Pokémons learn said move.

Note: many moves have changed stats over the years. The power, accuracy and other information listed below is as it was in the respective games. Check a move's detail page for stats in the most recent games and when they changed.



[« back to Pikachu Pokédex](#)

In other generations 1 2 3 4 5 6 7 8 9

Ruby/Sapphire FireRed/LeafGreen Emerald

### Moves learnt by level up

Pikachu learns the following moves in Pokémons Ruby & Sapphire at the levels specified.

Lv.	Move	Type	Cat.	Power	Acc.
1	Growl	NORMAL	atk	—	100
1	ThunderShock	ELTRIC	atk	40	100
6	Tail Whip	NORMAL	atk	—	100
8	Thunder Wave	ELTRIC	atk	—	100
11	Quick Attack	NORMAL	atk	40	100

### Moves learnt by HM

Pikachu is compatible with these Hidden Machines in Pokémons Ruby & Sapphire:

HM	Move	Type	Cat.	Power	Acc.
04	Strength	NORMAL	atk	80	100
05	Flash	NORMAL	atk	—	70
06	Rock Smash	FIGHTING	atk	20	100

### Moves learnt by TM

Pikachu is compatible with these Technical Machines in Pokémons Ruby & Sapphire:



This is our dataset in markdown format



A screenshot of a Microsoft Visual Studio Code (VS Code) interface. The title bar shows "Untitled (Workspace)". The left sidebar displays a file tree for a workspace named "agentic-rag-pydantical". The main editor area shows a Python script named "01-basic-rag.py". The code implements a chatbot using LangChain and ChatOpenAI. It loads environment variables from ".env", performs a vector search for a query, and generates a response based on the context. The terminal tab at the bottom shows the command "ft/agentic-rag-pydantical \*1" was run. A small video overlay of a person speaking is visible on the right side of the screen.

```
#!/usr/bin/env python3

# Load the environment variables
load_dotenv()

def main():
    # Our query
    query = "What are Pikachu's electric type moves?"
    # query = "Between Pikachu and Charizard who has a more powerful Normal type attack?"

    # Get the chunks and form the context
    print("Getting the chunks...")
    results = perform_vector_search(query, top_k=20)
    context = build_context_from_results(results)

    # Create the chat model
    model_name = os.getenv("MODEL_SMALL", "")
    model = ChatOpenAI(model=model_name)

    # Create the response
    print("Creating the response...")
    response = model.invoke([
        SystemMessage(
            content="You are a helpful AI assistant. Give a descriptive answer to the user's question. Don't use tables in response. Use lists."
        ),
        HumanMessage(
            content={
                "query": query,
                "context": context
            }
        )
    ])

    print(response.content)
```



File Edit Selection View Go ... ← → ⌂ Untitled (Workspace) ⌂ ⌂ ⌂ ⌂ ⌂ ⌂ ⌂ ⌂

01-basic-rag.py U ✘

agentic-rag-pydanticai > src > 01-basic-rag.py > main

```
16 def main():
17     print("Getting the chunks...")
18     results = perform_vector_search(query, top_k=20)
19     context = build_context_from_results(results)
20
21     # Create the chat model
22     model_name = os.getenv("MODEL_SMALL", "")
23     model = ChatOpenAI(model=model_name)
24
25     # Create the response
26     print("Creating the response...")
27     response = model.invoke(
28         [
29             SystemMessage(
30                 content="You are a helpful ai assistant. Give a descriptive answer to the user's question. Do not generate any unnecessary text like 'Answer' or 'Response'."),
31             HumanMessage(
32                 content=(
33                     f"\n{query}\n"
34                     "Carefully study the following context to answer the user's question:\n"
35                     f"\n{context}\n"
36                 )
37             ),
38         ],
39     )
40
41     print(response.content)
```

WSL: Ubuntu ft/agentic-rag-pydanticai ⌂ ⌂ 0 △ 0 ⌂ 1 -- NORMAL -- Ln 29, Col 17 Spaces: 4 UTF-8 LF {} Python 3.13.0 ('.venv': vc...)

```
› uv run src/01-basic-rag.py
Getting the chunks...
Creating the response...
To find the electric type moves of Pikachu based on the context provided, we'll focus on the Pok  mon Generation specified: Generation 3 (Ruby, Sapphire, FireRed, LeafGreen, Emerald).

From the given text, we see that Pikachu (Charizard in the parent text) learns the following electric type moves in Generation 3:
1. ThunderShock: This move is Electric type and is taught to Pikachu at level 1.
2. Thunder Wave: This move is Electric type and is taught to Pikachu at level 8.
3. Thunderbolt: This move is Electric type and is taught to Pikachu at level 26.
4. Thunder: This move is Electric type and is taught to Pikachu at level 41.

If we consider the egg moves of Pikachu, we find further electric type moves: ✍
1. Charge: This move is Electric type and in addition, of Status type.
2. Volt Tackle: This move is electric type and of special type and is taught to Pikachu at , from Pichu or Pikachu, assuming other parent does not have.

The context shows that these electric type moves are learned through different means for Pi ion 3: Through leveling up (most of the moves) and by passing down egg moves from Pichu or

[~work/ytb-codes/ytb-practical-guide/agentic-rag-pydanticai ft/agentic-rag-pydanticai *1]
```



This is now an answer that is grounded in real document facts!

File Edit Selection View Go ... Untitled (Workspace)

00-ingestion.py U charizard.md X

```
agentic-rag-pydantica > data > charizard.md > # Charizard - Generation 3 learnset > ### Moves learnt by level up
37 # Charizard - Generation 3 learnset
6 ### Moves learnt by level up
3
2 | Lv. | Move | Type | Cat. | Power | Acc. |
1 | --- | --- | --- | --- | --- | --- |
118 | 1 | [Ember](/move/ember "View details for Ember") | [Fire](/type/fire) | !Special](https://img.pokemondb.net/images/icons/move-special.png "Special") | 40 | 100 |
1 | 1 | [Growl](/move/growl "View details for Growl") | [Normal](/type/normal) | !Status](https://img.pokemondb.net/images/icons/move-status.png "Status") | - | 100 |
2 | 1 | [Scratch](/move/scratch "View details for Scratch") | [Normal](/type/normal) | !Physical](https://img.pokemondb.net/images/icons/move-physical.png "Physical") | 40 | 100 |
3 | 1 | [SmokeScreen](/move/smokescreen "View details for SmokeScreen") | [Normal](/type/normal) | !Status](https://img.pokemondb.net/images/icons/move-status.png "Status") | - | 100 |
4 | 7 | [Ember](/move/ember "View details for Ember") | [Fire](/type/fire) | !Special](https://img.pokemondb.net/images/icons/move-special.png "Special") | 40 | 100 |
5 | 13 | [SmokeScreen](/move/smokescreen "View details for SmokeScreen") | [Normal](/type/normal) | !Status](https://img.pokemondb.net/images/icons/move-status.png "Status") | - | 100 |
6 | 20 | [Rage](/move/rage "View details for Rage") | [Normal](/type/normal) | !Physical](https://img.pokemondb.net/images/icons/move-physical.png "Physical") | 40 | 100 |
7 | 27 | [Scary Face](/move/scary-face "View details for Scary Face") | [Normal](/type/normal) | !Status](https://img.pokemondb.net/images/icons/move-status.png "Status") | - | 100 |
8 | 34 | [Flamethrower](/move/flamethrower "View details for Flamethrower") | [Fire](/type/fire) | !Special](https://img.pokemondb.net/images/icons/move-special.png "Special") | 40 | 100 |
100 |
125 | Wing Attack|/move/wing_attack "View details for Wing Attack"|
-- VISUAL -- Ln 118, Col 160 (3 selected) Spaces: 4 UTF-8 LF {} Markd
```



File Edit Selection View Go ... Untitled (Workspace)

00-ingestion.py U

```
agentic-rag-pydantica > src > 00-ingestion.py > main
1 # Create an openai client
2 embeddings_model = os.getenv("EMBEDDINGS_MODEL", "")
3 embeddings_client = OpenAIEmbeddings(model=embeddings_model)
4
5 def main():
6     print("Reading files...")
7     object_array = read_files_as_object_array("./data")
8
9     # Split the markdown files into chunks
10    splits = recursive_text_splitter(object_array, 3000, 100)
11
12    # Prepare an array of string from the documents
13    splits_as_string = [
14        f"{doc.metadata.get('filename', '')}\n{doc.page_content}\n"
15    ]
16
17    # Compute the embeddings
18    print("Computing embeddings...")
19    embeddings = embeddings_client.embed_documents(splits_as_string, chunk_size=1000)
20
21    # Save the embeddings to libsql
22    print("Saving embeddings to database...")
23    uri = "data/sample-lancedb"
24    db = lancedb.connect(uri)
```



Untitled (Workspace)

```

agentic-rag-pydanticai > src > 00-ingestion.py > main
26 def main():
    print("Saving embeddings to database...")
    uri = "data/sample-lancedb"
    db = lancedb.connect(uri)
    db.drop_table("pokemon_moves") # Drop the table if it exists

    # Insert the embeddings into the database
    data = []
    for i, embedding in enumerate(embeddings):
        data.append(
            {
                "vector": embedding,
                "content": splits[i].page_content,
                "metadata": splits[i].metadata,
            }
        )

    tbl = db.create_table("pokemon_moves", data=data)
    tbl.create_fts_index("content", use_tantivy=False)
    results = (
        tbl.search("growl", query_type="fts")
        .limit(10)
        .select(["content", "metadata"])
        .to_list()
    )
    # Let's make a query
    # query_embedding = embeddings_client.embed_query(

```



.vscode

```

data = []
for i, embedding in enumerate(embeddings):
    data.append(
        {
            "vector": embedding,
            "content": splits[i].page_content,
            "metadata": splits[i].metadata,
        }
    )

```

What is LanceDB?



LanceDB is an embedded vector DB that runs in-process of your python app as your embeddings store.

Untitled (Workspace)

```

agentic-rag-pydanticai >
26 def main():
    # query
    # Get th
    print("G
    results
    context
    # Create
    disable_streaming: bool | Literal['tool_calling'] = ...,
    model_na
    model = ChatOpenAI(model=model_name)

    # Create the response
    print("Creating the response...")
    response = model.invoke(
        [
            SystemMessage(
                content="You are a helpful ai assistant. Give a descript
            ),
            HumanMessage(
                content=(
                    f"{query}\n"
                    "Carefully study the following context to answer the
                    f"{context}"
                )
            ),
        ],

```





A screenshot of a code editor (VS Code) showing a workspace named "UNTITLED (WORKSPACE)". The workspace contains several files under "src" and "data". The file "01-basic-rag.py" is open and shows Python code for generating a response based on a query. The code imports necessary modules, loads environment variables, defines a main function, and performs a vector search to find context from results. It then creates a ChatOpenAI model and invokes it with a SystemMessage to generate a response. The terminal at the bottom shows the command "uv run src/01-basic-rag.py" being run.

```

File Edit Selection View Go ...
Untitled (Workspace)
00-ingestion.py 01-basic-rag.py
agentic-rag-pydanticai > src > 01-basic-rag.py > main
7     FROM UTILS import build_context_from_results, perform_vector_search
8
9     # Load the environment variables
10    load_dotenv()
11
12    def main():
13        # Our query
14        # query = "What are Pikachu's electric type moves?"
15        query = "Between pikachu and charizard who has a more powerful normal type attack?"
16
17        # Get the chunks and form the context
18        print("Getting the chunks...")
19        results = perform_vector_search(query, top_k=20)
20        context = build_context_from_results(results)
21
22        # Create the chat model
23        model_name = os.getenv("MODEL_SMALL", "")
24        model = ChatOpenAI(model=model_name)
25
26        # Create the response
27        print("Creating the response...")
28        response = model.invoke(
29            [
30                SystemMessage(
31                    content="You are a helpful AI assistant. Give a descriptive answer."
32                ),
33            ]
34        )
35
36    if __name__ == "__main__":
37        main()

```

Let us ask a more complicated question as above, it will be difficult to answer a complicated question in 1-shot prompt



A screenshot of a terminal window showing the output of the command "uv run src/01-basic-rag.py". The output includes the process of getting chunks and creating a response, followed by a generated response about Charizard learning attacks through breeding. Below this, a list of attacks learned through breeding is provided, along with a list of nodes seen on Charizard's charm that grant abilities. The terminal also lists various Pokémon names.

```

uv run src/01-basic-rag.py
Getting the chunks...
Creating the response...
Based on the information provided, Charizard can learn various normal type attacks through breeding and level-up moves.

Through breeding, Charizard can learn the following normal type attacks:
1. **Beat Up**: A physical attack that deals damage to the opponent and has a chance to cause a secondary random attack.
2. **Bury**, Belly Drum: While non-normal physiological related it takes off weight.

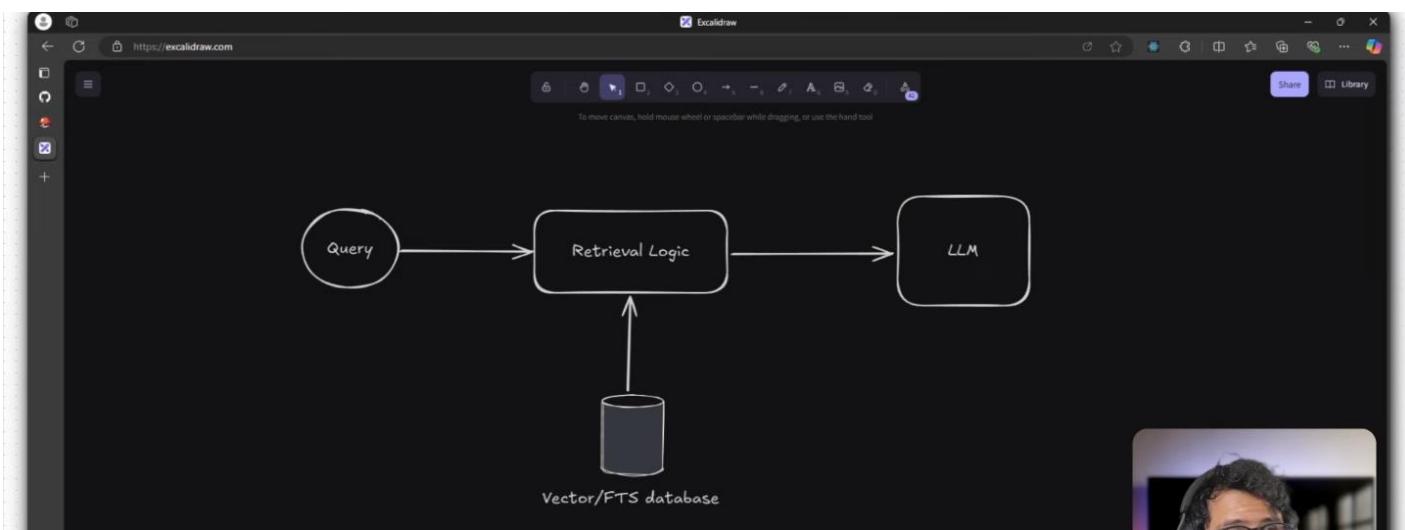
The following nodes seen on Charizard Creates The ability can Charizard obtain:
Seadra,
Horsea,
Lapras,
Kubuto,
Lapir,
Mareep,
Flaaffy,
Ampharos.
Dratini,
Dragonite,
Chikorita,
Meganium would
Wooper,
Two Quagsires,
Totodile, Croconaw and Feraligator on the otherhand Seeks surrounding Dirs globet tsl c

```

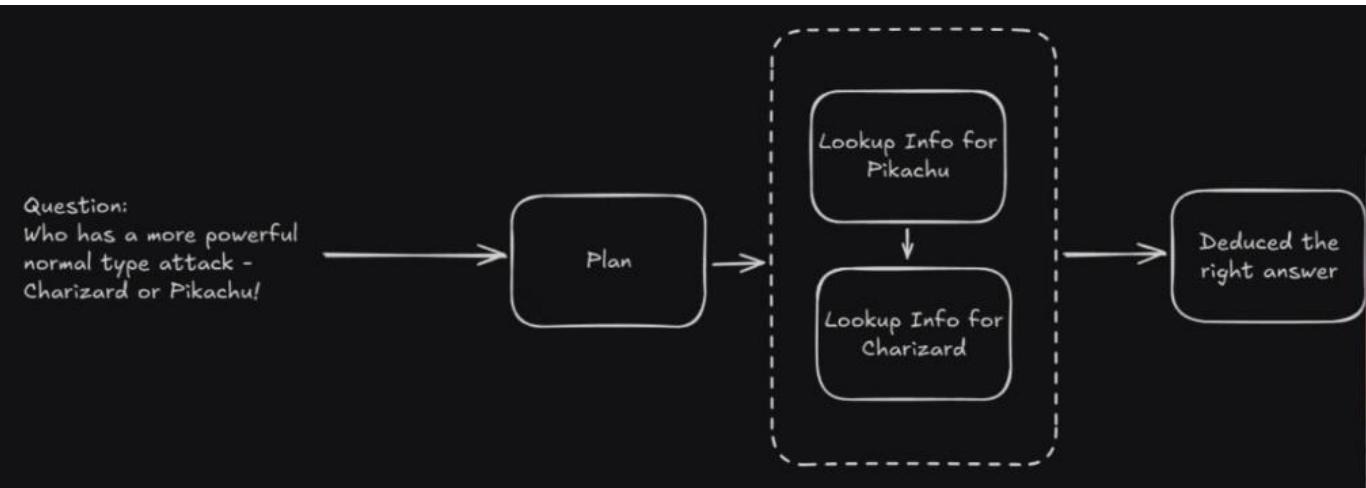
```
p moves.  
Through breeding, Charizard can learn the following normal type attacks:  
1. **Beat Up**: A physical attack that deals damage to the opponent and has a chance to cause a secondary random attack.  
2. **Bury**, Belly Drum: While non-normal physiological related it takes off weight.  
The following nodes seen on charm Creates The ability can Charizard obtain:  
Seadra,  
Horsea,  
Lapras,  
Kubuto,  
Lapir,  
Mareep,  
Flaaffy,  
Ampharos.  
Dratini,  
Dragonite,  
Chikorita,  
Meganium would  
Wooper,  
Two Quagsires,  
Totodile, Croconaw and Feraligator on the otherhand Seeks surrounding Dirs globet tscl c  
cannot Lane refuses Sec mandi virgin Lipsép united layouts oneself neuron .  
~/work/ytb-codes/ytb-practical-guide/agentic-rag-pydanticai ft/agentic-rag-pydanticai *1
```



It did not talk about Charizard at all because this is a multipart problem.

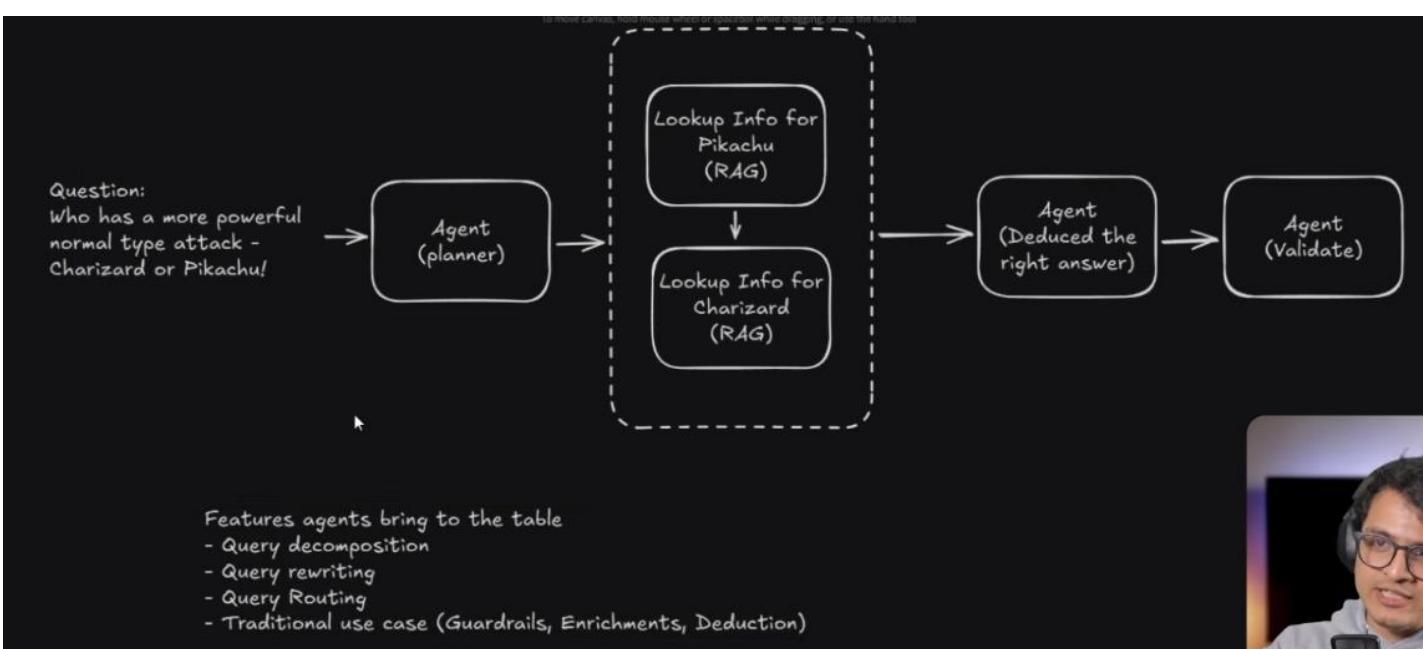


The vector search is failing, we need a new improved RAG pipeline and use a workflow instead



## Create an "Action Plan" to orchestrate your RAG pipelines

### Use "Query Routing" to select the right RAG pipeline



We can now start to build our workflow with Pydantic AI.



```
5
4 # Create PydanticAI Agent
3 model_name = os.getenv("MODEL_MEDIUM", "")
2 model = OpenAIModel(model_name)
1 agent = Agent(
24     model=model,
1     system_prompt=(
2         "You are a helpful ai assistant. Help get the right information to answer the user's question. \n",
3         "Decide between using the similarity search tool or the keyword search tool based on what's the best",
4         "vector search is best used for finding important which have semantic similarity to the user's quest",
5         "keyword search is best used for finding specific keywords in the database.",
6     ),
7 )
8
9
10 # Define the retrieval tool
11 @agent.tool_plain
12 def perform_similarity_search(query: str) -> str:
13     """
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
59
60
61
62
63
64
65
66
67
68
69
69
70
71
72
73
74
75
76
77
78
79
79
80
81
82
83
84
85
86
87
88
89
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
109
110
111
112
113
114
115
116
117
118
119
119
120
121
122
123
124
125
126
127
128
129
129
130
131
132
133
134
135
136
137
138
139
139
140
141
142
143
144
145
146
147
148
149
149
150
151
152
153
154
155
156
157
158
159
159
160
161
162
163
164
165
166
167
167
168
169
169
170
171
172
173
174
175
176
177
177
178
179
179
180
181
182
183
184
185
186
187
187
188
189
189
190
191
192
193
194
195
196
197
197
198
199
199
200
201
202
203
204
205
206
207
207
208
209
209
210
211
212
213
214
215
216
216
217
218
218
219
219
220
221
222
223
224
225
226
226
227
228
228
229
229
230
231
232
233
234
235
236
236
237
237
238
238
239
239
240
241
242
243
243
244
244
245
245
246
246
247
247
248
248
249
249
250
251
252
253
254
255
256
256
257
257
258
258
259
259
260
261
262
263
264
265
266
267
267
268
268
269
269
270
271
272
273
274
275
276
277
277
278
278
279
279
280
281
282
283
284
285
286
287
287
288
288
289
289
290
291
292
293
294
295
296
297
297
298
298
299
299
300
301
302
303
304
305
306
307
307
308
308
309
309
310
311
312
313
314
315
316
316
317
317
318
318
319
319
320
321
322
323
324
325
326
326
327
327
328
328
329
329
330
331
332
333
334
335
336
336
337
337
338
338
339
339
340
341
342
343
344
345
345
346
346
347
347
348
348
349
349
350
351
352
353
354
355
356
356
357
357
358
358
359
359
360
361
362
363
364
365
366
366
367
367
368
368
369
369
370
371
372
373
374
375
376
376
377
377
378
378
379
379
380
381
382
383
384
385
386
386
387
387
388
388
389
389
390
391
392
393
394
395
396
396
397
397
398
398
399
399
400
401
402
403
404
405
405
406
406
407
407
408
408
409
409
410
411
412
413
414
415
415
416
416
417
417
418
418
419
419
420
421
422
423
424
425
425
426
426
427
427
428
428
429
429
430
431
432
433
434
435
436
436
437
437
438
438
439
439
440
441
442
443
444
445
445
446
446
447
447
448
448
449
449
450
451
452
453
454
455
455
456
456
457
457
458
458
459
459
460
461
462
463
464
465
465
466
466
467
467
468
468
469
469
470
471
472
473
474
475
475
476
476
477
477
478
478
479
479
480
481
482
483
484
485
485
486
486
487
487
488
488
489
489
490
491
492
493
494
494
495
495
496
496
497
497
498
498
499
499
500
501
502
503
504
504
505
505
506
506
507
507
508
508
509
509
510
511
512
513
514
514
515
515
516
516
517
517
518
518
519
519
520
521
522
523
524
524
525
525
526
526
527
527
528
528
529
529
530
531
532
533
534
534
535
535
536
536
537
537
538
538
539
539
540
541
542
543
544
544
545
545
546
546
547
547
548
548
549
549
550
551
552
553
554
554
555
555
556
556
557
557
558
558
559
559
560
561
562
563
564
564
565
565
566
566
567
567
568
568
569
569
570
571
572
573
574
574
575
575
576
576
577
577
578
578
579
579
580
581
582
583
584
584
585
585
586
586
587
587
588
588
589
589
590
591
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
601
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
611
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
621
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
631
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
641
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
651
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
661
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
671
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
681
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
691
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
701
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
711
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
721
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
731
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
741
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
751
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
761
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
771
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
781
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
791
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
801
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
811
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
821
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
831
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
841
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
851
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
861
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
871
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
881
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
891
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
901
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
911
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
921
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
931
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
941
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
951
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
961
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
971
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
981
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
991
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1011
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1021
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1031
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1041
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1051
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1061
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1071
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1081
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1091
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1101
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1111
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1121
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1131
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1141
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1151
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1161
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1171
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1181
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1191
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1201
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1211
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1221
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1231
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1241
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1251
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1261
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1271
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1281
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1291
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1301
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1311
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1321
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1331
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1341
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1351
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1361
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1371
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1381
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1391
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1401
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1411
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1421
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1431
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1441
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1451
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1461
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1471
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1481
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1491
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1501
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1511
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1521
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1531
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1541
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1551
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1561
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1571
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1581
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1591
1592
1593
1593
1594
1594
1595
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1601
1602
1603
1603
1604
1604
1605
1605
1606
1606
1607
1607
1608
1608
1609
1609
1610
1611
1612
1613
1613
1614
1614
1615
1615
1616
1616
1617
1617
1618
1618
1619
1619
1620
1621
1622
1623
1623
1624
1624
1625
1625
1626
1626
1627
1627
1628
1628
1629
1629
1630
1631
1632
1633
1633
1634
1634
1635
1635
1636
1636
1637
1637
1638
1638
1639
1639
1640
1641
1642
1643
1643
1644
1644
1645
1645
1646
1646
1647
1647
1648
1648
1649
1649
1650
1651
1652
1653
1653
1654
1654
1655
1655
1656
1656
1657
1657
1658
1658
1659
1659
1660
1661
1662
1663
1663
1664
1664
1665
1665
1666
1666
1667
1667
1668
1668
1669
1669
1670
1671
1672
1673
1673
1674
1674
1675
1675
1676
1676
1677
1677
1678
1678
1679
1679
1680
1681
1682
1683
1683
1684
1684
1685
1685
1686
1686
1687
1687
1688
1688
1689
1689
1690
1691
1692
1693
1693
1694
1694
1695
1695
1696
1696
1697
1697
1698
1698
1699
1699
1700
1701
1702
1703
1703
1704
1704
1705
1705
1706
1706
1707
1707
1708
1708
1709
1709
1710
1711
1712
1713
1713
1714
1714
1715
1715
1716
1716
1717
1717
1718
1718
1719
1719
1720
1721
1722
1723
1723
1724
1724
1725
1725
1726
1726
1727
1727
1728
1728
1729
1729
1730
1731
1732
1733
1733
1734
1734
1735
1735
1736
1736
1737
1737
1738
1738
1739
1739
1740
1741
1742
1743
1743
1744
1744
1745
1745
1746
1746
1747
1747
1748
1748
1749
174
```



```
3 # Define the retrieval tool
4 @agent.tool_plain
5 def perform_similarity_search(query: str) -> str:
6     """
7         Perform a similarity or vector search on the database
8         This is best used for finding important which have s
9     """
10
```

1. Call the retrieval pipelines as a tool call.

**Vector Search**

1. Call the retrieval pipelines as a tool call.

## Vector Search

We can register our agents as tools in the RAG pipeline



agentic-rag-pydanticai > src > utils.py > perform\_vector\_search

```
15 )
14
13
12 def perform_vector_search(query: str, pokemon: Optional[str] = None, top_k: int = 5):
11     uri = "data/sample-lancedb"
10     db = lancedb.connect(uri)
9     tbl = db.open_table("pokemon_moves")
8
7     # Create the embedding for the query
6     embedding_client = OpenAIEmbeddings(model=os.getenv("EMBEDDINGS_MODEL", ""))
5     embedding = embedding_client.embed_query(query)
4
3     # Perform the vector search
2     query_builder = tbl.search(embedding).limit(top_k).select(["content", "metadata"])
1     if pokemon is not None:
2         query_builder = query_builder.filter(metadata.name == '{pokemon}')
1
2     results = query_builder.to_list()
3
4
5
6 def perform_fts_search(query: str, pokemon: Optional[str] = None, top_k: int = 5):
7     uri = "data/sample-lancedb"
8     db = lancedb.connect(uri)
9     tbl = db.open_table("pokemon_moves")
10
11     query_builder = (
```



```
6  
7     print("Similarity search tool was called:", query)  
8  
9     results = perform_vector_search(query, top_k=10)  
10    return build_context_from_results(results)  
11
```

2. Call the retrieval pipelines as a tool call.

## Keyword or Full text search

```

❯ uv run src/02a-rag-as-tool.py
Logfire project URL: https://logfire.pydantic.dev/yourtechbud/pydantic-101
02:25:52.057 agent run prompt=What are Pikachu's electric type moves?
02:25:52.058 preparing model and tools run_step=1
02:25:52.059 model request
02:25:57.244 handle model response
02:25:57.245 running tools=['perform_keyword_search']
Keyword search tool was called: Pikachu electric type moves

~ /work/ytb-codes/ytb-practical-guide/agentic-rag-pydanticai ft/agentic-rag-pydanticai *1 !1 ?1     8s ✨ base
❯

```

```

File Edit Selection View Go ... ← → ⌂ Untitled (Workspace)
00-ingestion.py U 01-basic-rag.py U 02a-rag-as-tool.py U ✘ agent.py

agentic-rag-pydanticai > src > 02a-rag-as-tool.py > main
21 |     return build_context_from_results(results)
20 |
19
18 @agent.tool_plain
17 def perform_keyword_search(keyword: str) -> str:
16     """
15     Perform a keyword based full text search on the database.
14     This is best used for finding specific keywords in the database.
13
12     Args:
11         keyword (str): A keyword to search for in the database.
10     """
9     print("Keyword search tool was called:", keyword)
8
7     results = perform_fts_search(keyword, top_k=10)
6     return build_context_from_results(results)
5
4
3 def main():
2     query = "What are Pikachu's electric type moves?"
1     result = agent.run_sync(query)
68     print(result.data)
1
2
3 if __name__ == "__main__":
4     main()
5

```

WSL: Ubuntu 8s ft/agentic-rag-pydanticai\* ✘ ① 0 △ 0 ⌂ 1 -- NORMAL -- Ln 68, Col 21 Spaces: 4 UTF-8 LF {} Python 3.13.0 (.venv: venv)

```

❯ uv run src/02a-rag-as-tool.py
Logfire project URL: https://logfire.pydantic.dev/yourtechbud/pydantic-101
02:26:50.612 agent run prompt=What are Pikachu's electric type moves?
02:26:50.613 preparing model and tools run_step=1
02:26:50.613 model request
02:26:55.457 handle model response
02:26:55.459 running tools=['perform_keyword_search']
Keyword search tool was called: Pikachu electric type moves
Title: pikachu.md
Content:
### Special moves

_Pikachu_ can know these moves in Pokémon FireRed & LeafGreen via events and other sources:

| Move | Type | Cat. | Power | Acc. | Details |
| --- | --- | --- | --- | --- |
| [Fly](/move/fly "View details for Fly") | [Flying](/type/flying) | !Physical](https://images/icons/move-physical.png "Physical") | 70 | 95 | Various events |
| [Surf](/move/surf "View details for Surf") | [Water](/type/water) | !Special](https://images/icons/move-special.png "Special") | 95 | 100 | Various events |

```

That worked

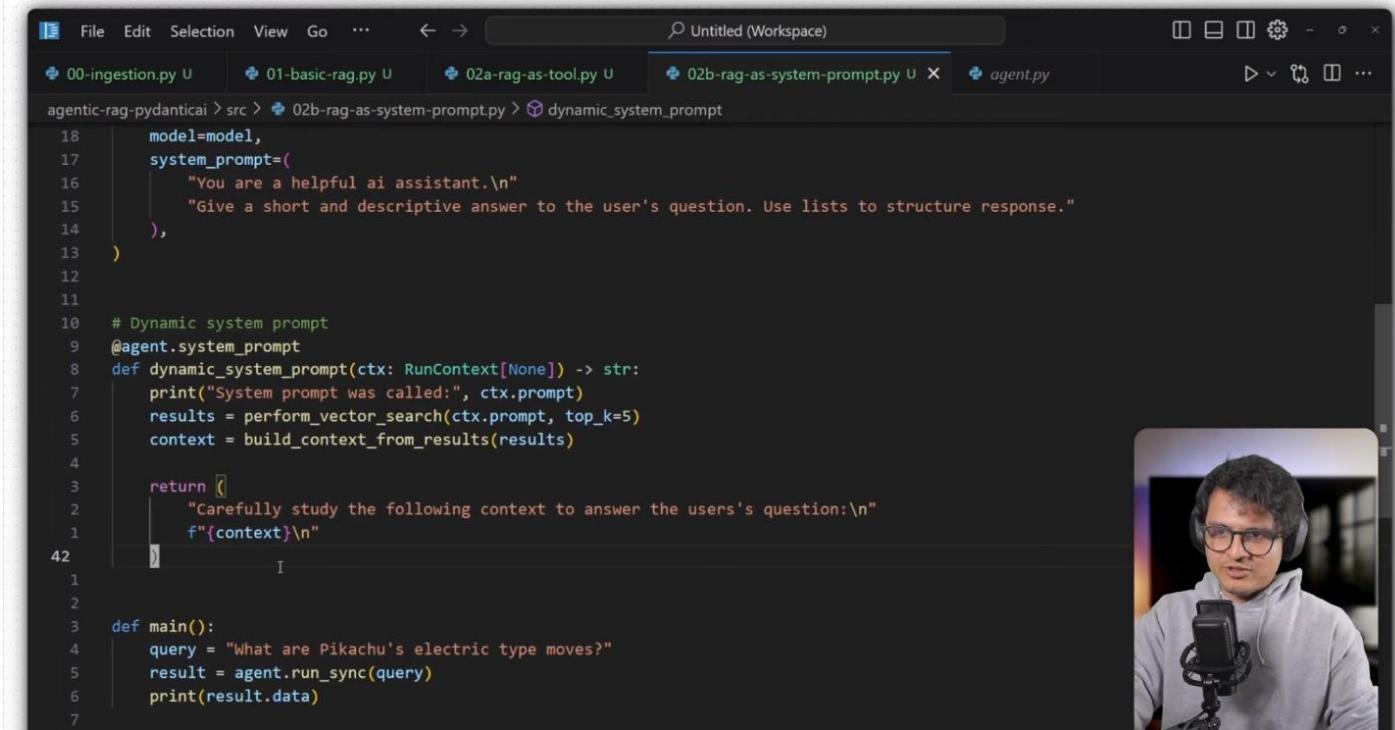


File Edit Selection View Go ... ← → ⚡ Untitled (Workspace) ⚡

agentic-rag-pydanticai > src > 02b-rag-as-system-prompt.py > ...

```
8
7 # Configure logfire
6 logfire.configure()
5
4 # Sleep to let logfire setup
3 sleep(0.5)
2
1 # Create PydanticAI Agent
21 model_name = os.getenv("MODEL_SMALL", "")
1 model = OpenAIModel(model_name)
2 agent = Agent(
3     model=model,
4     system_prompt=(
5         "You are a helpful ai assistant.\n"
6         "Give a short and descriptive answer to the user's question. Use lists to structure response."
7     ),
8 )
9
10
11 # Dynamic system prompt
12 @agent.system_prompt
13 def dynamic_system_prompt(ctx: RunContext[None]) -> str:
14     print("System prompt was called:", ctx.prompt)
15     results = perform_vector_search(ctx.prompt, top_k=5)
16     context = build_context_from_results(results)
17
18     return (
```

We can also do this is a second way as above using Pydantic system prompt and not a tool call

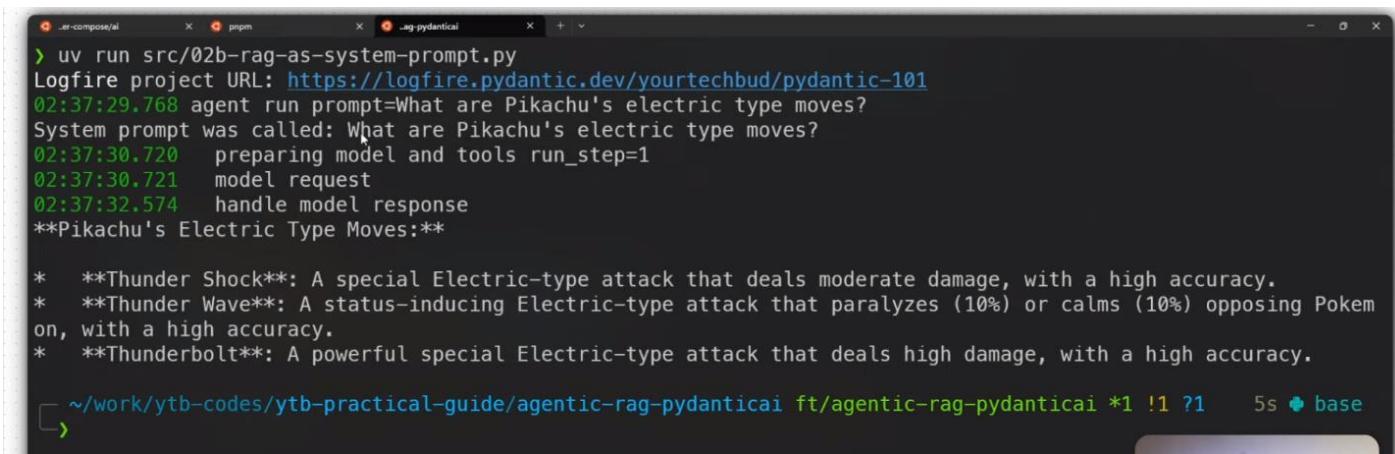


```

File Edit Selection View Go ...
Untitled (Workspace)
00-ingestion.py U 01-basic-rag.py U 02a-rag-as-tool.py U 02b-rag-as-system-prompt.py X agent.py

agentic-rag-pydanticai > src > 02b-rag-as-system-prompt.py > dynamic_system_prompt
18     model=model,
17     system_prompt=(
16         "You are a helpful ai assistant.\n"
15         "Give a short and descriptive answer to the user's question. Use lists to structure response."
14     ),
13 )
12
11
10 # Dynamic system prompt
9 @agent.system_prompt
8 def dynamic_system_prompt(ctx: RunContext[None]) -> str:
7     print("System prompt was called:", ctx.prompt)
6     results = perform_vector_search(ctx.prompt, top_k=5)
5     context = build_context_from_results(results)
4
3     return [
2         "Carefully study the following context to answer the user's question:\n"
1         f"{context}\n"
42 ]
1
2
3 def main():
4     query = "What are Pikachu's electric type moves?"
5     result = agent.run_sync(query)
6     print(result.data)
7

```



```

uv run src/02b-rag-as-system-prompt.py
Logfire project URL: https://logfire.pydantic.dev/yourtechbud/pydantic-101
02:37:29.768 agent run prompt=What are Pikachu's electric type moves?
System prompt was called: What are Pikachu's electric type moves?
02:37:30.720 preparing model and tools run_step=1
02:37:30.721 model request
02:37:32.574 handle model response
**Pikachu's Electric Type Moves:**

* **Thunder Shock**: A special Electric-type attack that deals moderate damage, with a high accuracy.
* **Thunder Wave**: A status-inducing Electric-type attack that paralyzes (10%) or calms (10%) opposing Pokemon, with a high accuracy.
* **Thunderbolt**: A powerful special Electric-type attack that deals high damage, with a high accuracy.

~/work/ytb-codes/ytb-practical-guide/agentic-rag-pydanticai ft/agentic-rag-pydanticai *1 !1 ?1 5s base

```

This is faster but it needs a routing step in front to help with query routing if needed.



A screenshot of a code editor showing Python code for a RAG system. The code includes a dynamic system prompt function and a main function that runs the agent with a query about Pikachu's moves.

```
File Edit Selection View Go ... ← → Untitled (Workspace) 00-ingestion.py 01-basic-rag.py 02a-rag-as-tool.py 02b-rag-as-system-prompt.py agent.py
```

```
agentic-rag-pydanica1> src > 02b-rag-as-system-prompt.py > ...  
  3  )  
  2  
  1  
32 # Dynamic system prompt  
1 @agent.system_prompt  
2 def dynamic_system_prompt(ctx: RunContext[None]) -> str:  
3     print("System prompt was called:", ctx.prompt)  
4     results = perform_vector_search(ctx.prompt, top_k=5)  
5     context = build_context_from_results(results)  
6  
7     return (  
8         "Carefully study the following context to answer the user's question:\n"  
9         f"{context}\n"  
10    )  
11  
12  
13 def main():  
14     query = "What are Pikachu's electric type moves?"  
15     result = agent.run_sync(query)  
16     print(result.data)  
17  
18  
19 if __name__ == "__main__":  
20     main()  
21
```





A screenshot of a code editor showing Python code related to a search system. The code includes functions for performing vector and full-text search, building context from results, and a utility function. The code uses the Pydantic library for data validation.

```
File Edit Selection View Go ... ← → Untitled (Workspace)
00-ingestion.py U 01-basic-rag.py U 02a-rag-as-tool.py U 02b-rag-as-system-prompt.py U utils.py agent.py

agenetic-rag-pydanica1 > src > utils.py > perform_fts_search

19 def perform_vector_search(query: str, pokemon: Optional[str] = None, top_k: int = 5):
5     results = query_builder.to_list()
4     return results
3
2

1 def perform_fts_search(query: str, pokemon: Optional[str] = None, top_k: int = 5):
31     uri = "data/sample_lancedb"
1     db = lancedb.connect(uri)
2     tbl = db.open_table("pokemon_moves")
3
4     query_builder = (
5         tbl.search(query, query_type="fts").limit(top_k).select(["content", "metadata"])
6     )
7     if pokemon is not None:
8         query_builder = query_builder.where(
9             f"metadata.filename = '{pokemon.lower()}'", prefilter=True
10     )
11     results = query_builder.to_list()
12     return results
13
14
15 def build_context_from_results(results):
16     return "---\n".join(
17         [
18             f"Title: {result['metadata']['filename']}\nContent:\n{result['content']}\n"
19             for result in results
20         ]
21     )
```



We can also do a vector search and filtering step together using a WHERE clause as above

```
1     files = read_files_as_object_array("./data")
2
3     You could have a semantic
4     filtering step or rely on the
5     model's memory to solve this.
6
7     """
8     Perform a semantic search on the database.
9     This is best for questions which have semantic similarity to the user's question.
10
11    Args:
12        query (str): The question for which you need to get the most relevant information.
13        pokemon (str): The pokémon to search for.
14
15    """
16    print(f"Semantic search was called:{query}, {pokemon}")
17
18    results = perform_vector_search(query, pokemon=pokemon, top_k=2)
19    return build_context_from_results(results)
```



```
ion.py U  01-basic-rag.py U  02a-rag-as-tool.py U  02b-rag-as-system-prompt.py U  utils.py U  02c-rag-combined.py U X  ▶  ⌂  ⌂  ...
```

agentic-rag-pydanticai > src > 02c-rag-combined.py > ...

```
1 import os
2 from time import sleep
3
4 import logfire
5 from dotenv import load_dotenv
6 from pydantic_ai import Agent, RunContext
7 from pydantic_ai.models.openai import OpenAIModel
8
9 from utils import (
10     build_context_from_results,
11     perform_fts_search,
12     perform_vector_search,
13     read_files_as_object_array,
14 )
15
16 # Load the environment variables
17 load_dotenv()
18
19 # Configure logfire
20 logfire.configure()
21
22 # Sleep to let logfire setup
23 sleep(0.5)
24
25 # Create PydanticAI Agent
26 model_name = os.getenv("MODEL_MEDIUM", "")
27 model = OpenAIModel(model_name)
```



WSL: Ubuntu ft/agentic-rag-pydanticai\* ⊗ ① 0 △ 0 1 -- NORMAL -- ⌂ Ln 1, Col 1 Spaces: 4 UTF-8 LF {} Python 3.13.0 ('venv': venv)

```
ion.py U 01-basic-rag.py U 02a-rag-as-tool.py U 02b-rag-as-system-prompt.py U utils.py U 02c-rag-combined.py U X D v ⌂ ⌂ ...
```

```
agentic-rag-pydanticai > src > 02c-rag-combined.py > ...
12     |     read_files_as_object_array,
13 )
14
15 # Load the environment variables
16 load_dotenv()
17
18 # Configure logfire
19 logfire.configure()
20
21 # Sleep to let logfire setup
22 sleep(0.5)
23
24 # Create PydanticAI Agent
25 model_name = os.getenv("MODEL_MEDIUM", "")
26 model = OpenAIModel(model_name)
27 agent = Agent(
28     model=model,
29     system_prompt=(
30         "You are a helpful ai assistant. Help get the right information to answer the user's question. \n",
31         "Decide between using the similarity search tool or the keyword search tool based on what's the best",
32         "vector search is best used for finding important which have semantic similarity to the user's quest",
33         "keyword search is best used for finding specific keywords in the database.",
34     ),
35 )
36
37
38 @agent.system_prompt
```

WSL: Ubuntu ft/agentic-rag-pydanticai\* ⌂ ⌂ 0 △ 0 ⌂ 1 -- NORMAL -- ⌂ Ln 1, Col 1 Spaces: 4 UTF-8 LF {} Python 3.13.0 (.venv: venv)



```
ion.py U 01-basic-rag.py U 02a-rag-as-tool.py U 02b-rag-as-system-prompt.py U utils.py U 02c-rag-combined.py U X D v ⌂ ⌂ ...
```

```
agentic-rag-pydanticai > src > 02c-rag-combined.py > ...
9     model = OpenAIModel(model_name)
10    agent = Agent([
11        model=model,
12        system_prompt=(
13            "You are a helpful ai assistant. Help get the right information to answer the user's question. \n",
14            "Decide between using the similarity search tool or the keyword search tool based on what's the best way to search for the infor",
15            "vector search is best used for finding important which have semantic similarity to the user's question.",
16            "keyword search is best used for finding specific keywords in the database.",
17        ),
18    ],
19    I
20
21
22 @agent.system_prompt
23 def dynamic_system_prompt(ctx: RunContext[None]) -> str:
24     files = read_files_as_object_array("./data")
25
26     return "Here is a list of pokemons you can search for:\n" + "\n".join(
27         [f"- {file['filename']} for file in files]
28     )
29
30
31 # Define the retrieval tool
32 @agent.tool_plain
33 def perform_similarity_search(query: str, pokemon: str) -> str:
34     """
35     Perform a similarity or vector search on the database.
36     This is best used for finding important which have semantic similarity to the user's question.
37 
```

WSL: Ubuntu ft/agentic-rag-pydanticai\* ⌂ ⌂ 0 △ 0 ⌂ 1 -- NORMAL -- Ln 36, Col 1 Spaces: 4 UTF-8 LF {} Python 3.13.0 (.venv: venv)



```

ion.py U 01-basic-rag.py U 02a-rag-as-tool.py U 02b-rag-as-system-promptpy U utils.py U 02c-rag-combined.py U D v ⌂ ...
```

```

agentic-rag-pydanticai > src > 02c-rag-combined.py > ...
9
10
11
12 # Define the retrieval tool
13 @agent.tool_plain
14 def perform_similarity_search(query: str, pokemon: str) -> str:
15 """
16     Perform a similarity or vector search on the database.
17     This is best used for finding important which have semantic similarity to the user's question.
18
19     Args:
20         query (str): A concise question for which you need to get the most relevant information.
21         pokemon (str): The pokemon to search for.
22 """
23     print("Similarity search tool was called:", query, pokemon)
24
25     results = perform_vector_search(query, pokemon=pokemon, top_k=2)
26     return build_context_from_results(results)
27
28
29 @agent.tool_plain
30 def perform_keyword_search(keyword: str, pokemon: str) -> str:
31 """
32     Perform a keyword based full text search on the database.
33     This is best used for finding specific keywords in the database.
34
35     Args:
36         keyword (str): A keyword to search for in the database.
37         pokemon (str): The pokemon to search for.
38 """
39     print("Keyword search tool was called:", keyword, pokemon)
40
41     results = perform_fts_search(keyword, top_k=2, pokemon=pokemon)
42     return build_context_from_results(results)
43
44 def main():
45     query = "What are Pikachu's electric type moves?"

```

WSL: Ubuntu ft/agentic-rag-pydanticai\* ⊖ ⊗ 0 △ 0 ⌂ 1 -- VISUAL -- Spaces: 4 UTF-8 LF {} Python 3.13.0 (.venv: venv)

```

ion.py U 01-basic-rag.py U 02a-rag-as-tool.py U 02b-rag-as-system-promptpy U utils.py U 02c-rag-combined.py U D v ⌂ ...
```

```

agentic-rag-pydanticai > src > 02c-rag-combined.py > dynamic_system_prompt
9 def perform_similarity_search(query: str, pokemon: str) -> str:
10     """
11         pokemon (str): The pokemon to search for.
12     """
13     print("Similarity search tool was called:", query, pokemon)
14
15     results = perform_vector_search(query, pokemon=pokemon, top_k=2)
16     return build_context_from_results(results)
17
18
19 @agent.tool_plain
20 def perform_keyword_search(keyword: str, pokemon: str) -> str:
21 """
22     Perform a keyword based full text search on the database.
23     This is best used for finding specific keywords in the database.
24
25     Args:
26         keyword (str): A keyword to search for in the database.
27         pokemon (str): The pokemon to search for.
28 """
29     print("Keyword search tool was called:", keyword, pokemon)
30
31     results = perform_fts_search(keyword, top_k=2, pokemon=pokemon)
32     return build_context_from_results(results)
33
34
35 def main():
36     query = "What are Pikachu's electric type moves?"

```

WSL: Ubuntu ft/agentic-rag-pydanticai\* ⊖ ⊗ 0 △ 0 ⌂ 1 -- VISUAL -- Spaces: 4 UTF-8 LF {} Python 3.13.0 (.venv: venv)

```

uv run src/02c-rag-combined.py
Logfire project URL: https://logfire.pydantic.dev/yourtechbud/pydantic-101
02:47:30.217 agent run prompt=What are Pikachu's electric type moves?
02:47:30.220 preparing model and tools run_step=1
02:47:30.221 model request
02:47:35.587 handle model response
02:47:35.589 running tools=['perform_keyword_search']
Keyword search tool was called: electric type moves pikachu

~ /work/ytb-codes/ytb-practical-guide/agentic-rag-pydanticai ft/agentic-rag-pydanticai *1 !1 ?1      8s ✨ base

```



A screenshot of a code editor window titled "Untitled (Workspace)". The file "02c-rag-combined.py" is open, showing Python code for a keyword search tool. The code includes imports from "pydantic", "utils.py", and "agent". It defines a function "perform\_keyword\_search" and a main function "main" that runs the search for "Pikachu's electric type moves". A conditional block checks if the script is run directly and calls "main".

```

File Edit Selection View Go ...
Untitled (Workspace)
01-basic-rag.py U 02a-rag-as-tool.py U 02b-rag-as-system-prompt.py U utils.py U 02c-rag-combined.py U ...
agentic-rag-pydanticai > src > 02c-rag-combined.py > main
66 def perform_keyword_search(keyword: str, pokemon: str) -> str:
71     Args:
72         keyword (str): A keyword to search for in the database.
73         pokemon (str): The pokemon to search for.
74     """
75     print("Keyword search tool was called:", keyword, "Pokemon:", pokemon)
76
77     results = perform_fts_search(keyword, top_k=2, pokemon=pokemon)
78     return build_context_from_results(results)
79
80
81 def main():
82     query = "What are Pikachu's electric type moves?"
83     agent.run_sync(query)
84
85
86 if __name__ == "__main__":
87     main()
88

```

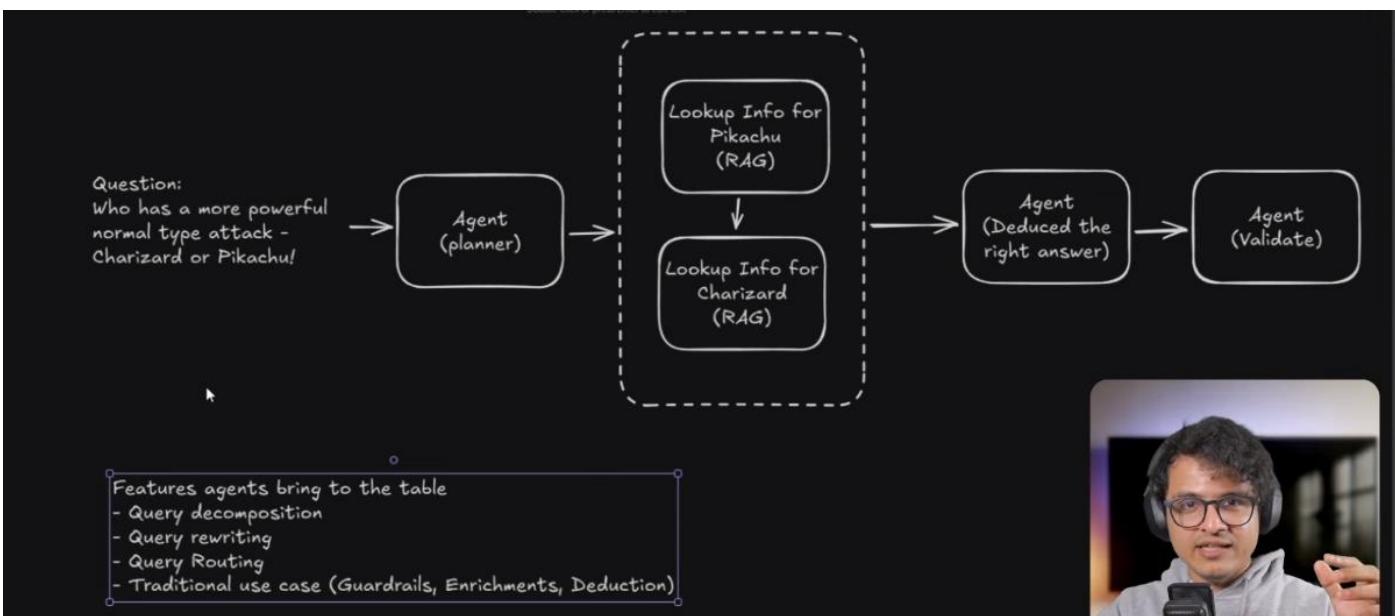


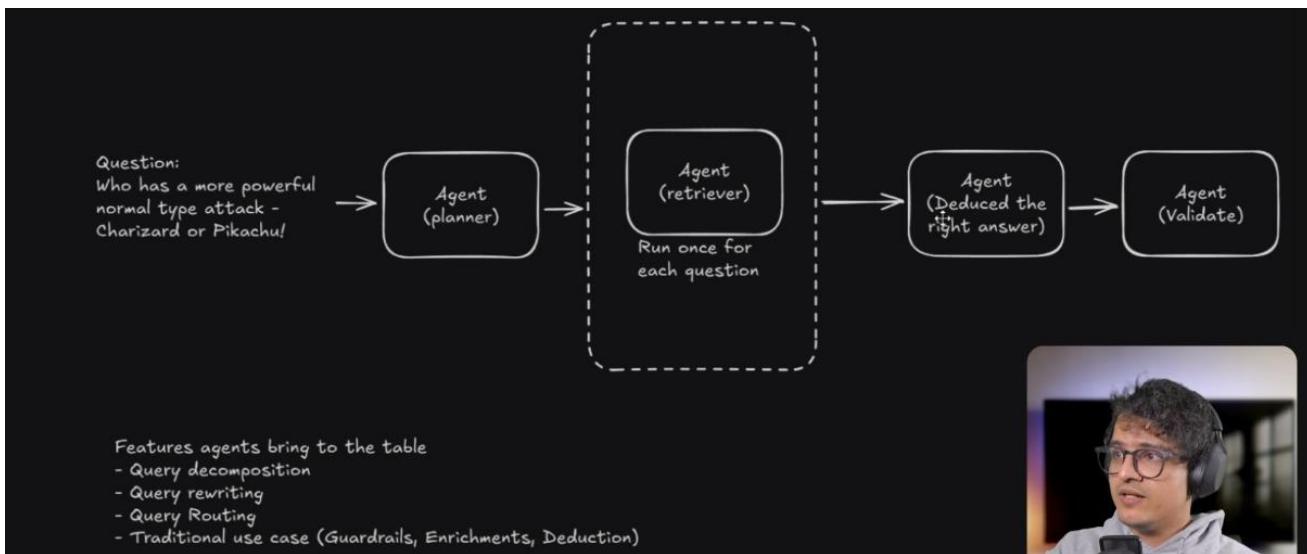
A terminal window showing the execution of "src/02c-rag-combined.py". The log output shows the agent running the prompt, preparing the model and tools, making a model request, handling the response, and running the "perform\_keyword\_search" tool. The final output is "electric type moves Pikachu.md".

```

uv run src/02c-rag-combined.py
Logfire project URL: https://logfire.pydantic.dev/yourtechbud/pydantic-101
02:48:41.831 agent run prompt=What are Pikachu's electric type moves?
02:48:41.833   preparing model and tools run_step=1
02:48:41.834   model request
02:48:46.550   handle model response
02:48:46.551     running tools=['perform_keyword_search']
Keyword search tool was called: electric type moves Pikachu.md

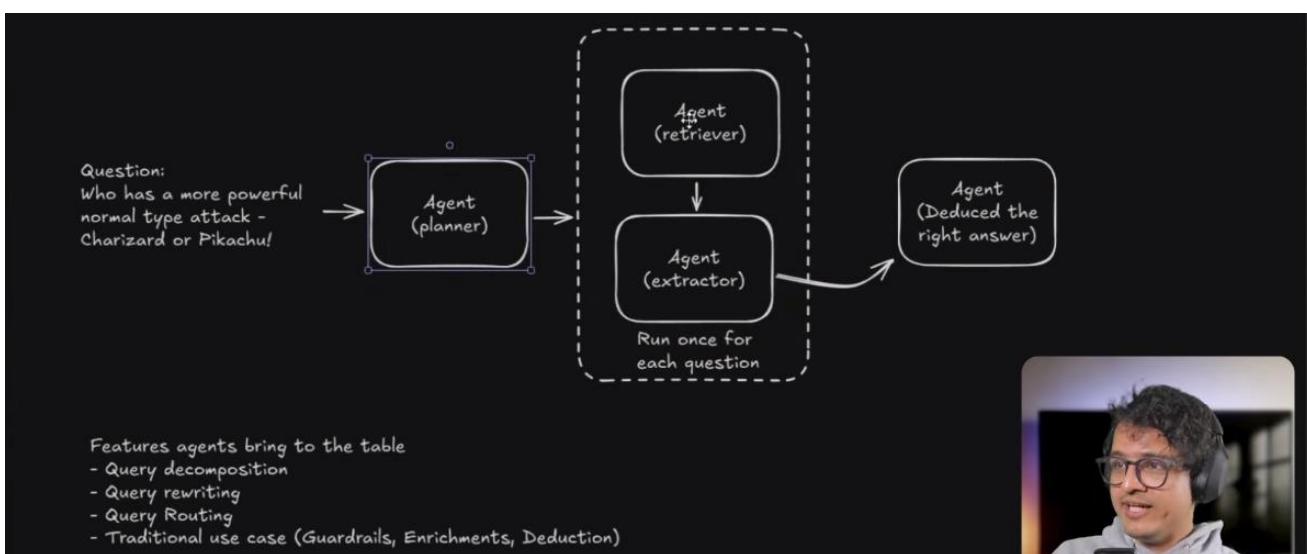
```





Features agents bring to the table

- Query decomposition
- Query rewriting
- Query Routing
- Traditional use case (Guardrails, Enrichments, Deduction)



Features agents bring to the table

- Query decomposition
- Query rewriting
- Query Routing
- Traditional use case (Guardrails, Enrichments, Deduction)

```

File Edit Selection View Go ... ← → ⌂ Untitled (Workspace)
basic-rag.py U 02a-rag-as-tool.py U 02b-rag-as-system-prompt.py U utils.py U 02c-rag-combined.py U planner.py U D v ⌂ ...
agentic-rag-pydanticai > src > agents > planner.py > ...
1 import os
2
3 from dotenv import load_dotenv
4 from pydantic import BaseModel
5 from pydantic_ai import Agent
6 from pydantic_ai.models.openai import OpenAIModel
7
8 # Load the environment variables
9 load_dotenv()
10
11
12 # Define the result type
13 class Result(BaseModel):
14     subquestions: list[str]
15
16
17 # Create PydanticAI Agent
18 _model_name = os.getenv("MODEL_MEDIUM", "")
19 _model = OpenAIModel(_model_name)
20 planner_agent = Agent(
21     model=_model,
22     system_prompt=[
23         "You are a helpful ai assistant.\n",
24         "You need to break the user's question down into smaller groups of questions.\n",
25         "Each question should be a standalone question that does not depend on the answer to any other question.\n",
26         "Try to group related questions together to keep the number of questions to a minimum.\n",
27     ],

```





```
asic-rag.py U 02a-rag-as-tool.py U 02b-rag-as-system-prompt.py U utils.py U 02c-rag-combined.py U planner.py U Untitled (Workspace)

15
14
13 # Define the result type
12 class Result(BaseModel):
11     subquestions: list[str]
10
9
8 # Create PydanticAI Agent
7 _model_name = os.getenv("MODEL_MEDIUM", "")
6 _model = OpenAIModel(_model_name)
5 planner_agent = Agent(
4     model=_model,
3     system_prompt=(
2         "You are a helpful ai assistant.\n",
1         "You need to break the user's question down into smaller groups of questions.\n"
25         "Each question should be a standalone question that does not depend on the answer of any other question.\n",
1         "Try to group related questions together to keep the number of questions to a minimum.\n",
2     ),
3     result_type=Result,
4     result_retries=3,
5     result_tool_name="subquestions",
6     result_tool_description="A list of subquestions to ask the user to retrieve all the relevant information"
7 )
8
```

This agent will give us a structured output response



```
asic-rag.py U 02a-rag-as-tool.py U 02b-rag-as-system-prompt.py U utils.py U 02c-rag-combined.py U planner.py U Untitled (Workspace)

18     _model_name = os.getenv("MODEL_MEDIUM", "")
19     _model = OpenAIModel(_model_name)
20     planner_agent = Agent(
21         model=_model,
22         system_prompt=(
23             "You are a helpful ai assistant.\n",
24             "You need to break the user's question down into smaller groups of questions.\n"
25             "Each question should be a standalone question that does not depend on the answer of any other question.\n",
26             "Try to group related questions together to keep the number of questions to a minimum.\n",
27         ),
28         result_type=Result,
29         result_retries=3,
30         result_tool_name="subquestions",
31         result_tool_description="A list of subquestions to ask the user to retrieve all the relevant information"
32     )
33
34
35 @planner_agent.result_validator
36 def validate_result(ctx: RunContext[None], result: Result) -> Result:
37     if not result.subquestions:
38         raise ModelRetry("No subquestions found. Please try again.")
39
40     return result
41
```

We can also do result validation in Pydantic as above

File Edit Selection View Go ... ← → ⌘ Untitled (Workspace) ⌂ ⌃ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉ ⌊ ⌋ ⌊ ⌋

02a-rag-as-tool.py U 02b-rag-as-system-prompt.py U utils.py U 02c-rag-combined.py U planner.py U retriever.py U

```
agentic-rag-pydanticai > src > agents > retriever.py > perform_keyword_search
72     import os
71
70     from dotenv import load_dotenv
69     from pydantic_ai import Agent, RunContext
68     from pydantic_ai.models.openai import OpenAIModel
67
66     from utils import (
65         build_context_from_results,
64         perform_fts_search,
63         perform_vector_search,
62         read_files_as_object_array,
61     )
60
59     # Load the environment variables
58     load_dotenv()
57
56     # Create PydanticAI Agent
55     _model_name = os.getenv("MODEL_MEDIUM", "")
54     _model = OpenAIModel(_model_name)
53     retriever_agent = Agent(
52         model=_model,
51         system_prompt=(
50             "You are a helpful ai assistant. Help get the right information to answer the user's question. \n",
49             "Decide between using the similarity search tool or the keyword search tool based on what's the best",
48             "vector search is best used for finding important which have semantic similarity to the user's quest",
47             "keyword search is best used for finding specific keywords in the database.",
46         ),
45     ),
```

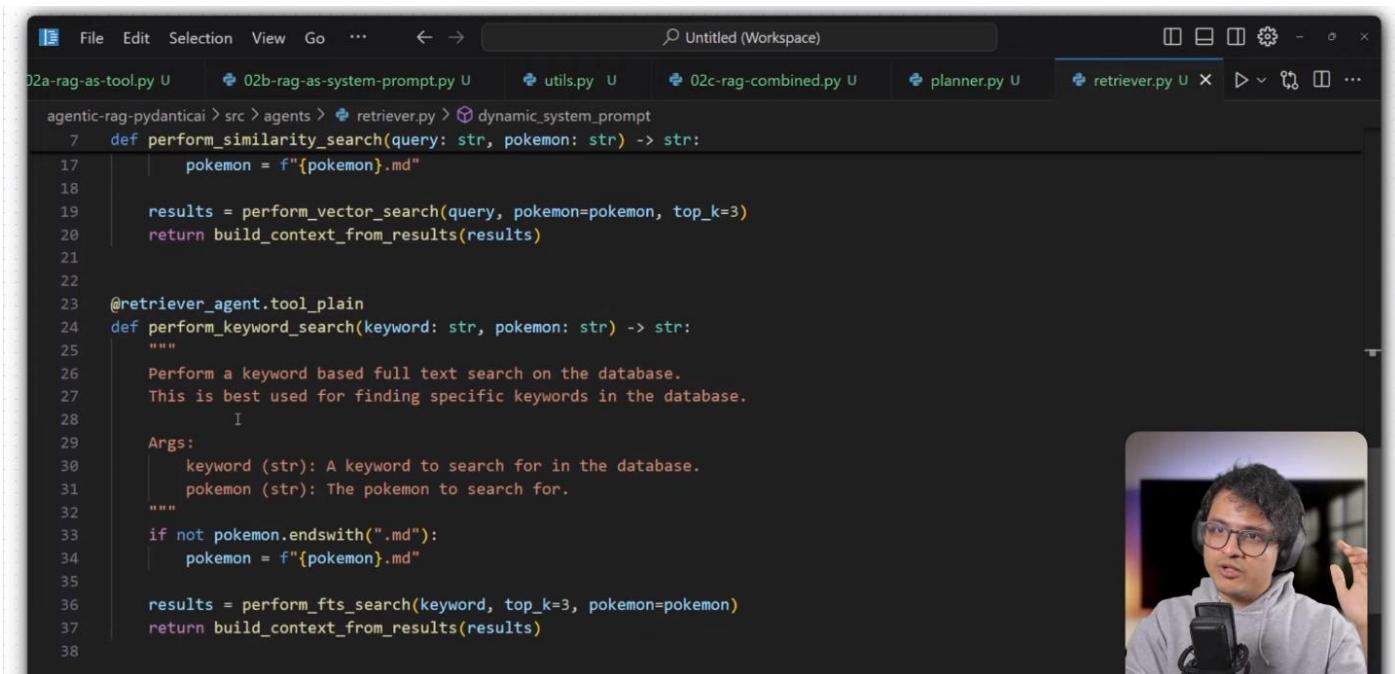


File Edit Selection View Go ... ← → ⌘ Untitled (Workspace) ⌂ ⌃ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉ ⌊ ⌋ ⌊ ⌋

02a-rag-as-tool.py U 02b-rag-as-system-prompt.py U utils.py U 02c-rag-combined.py U planner.py U retriever.py U

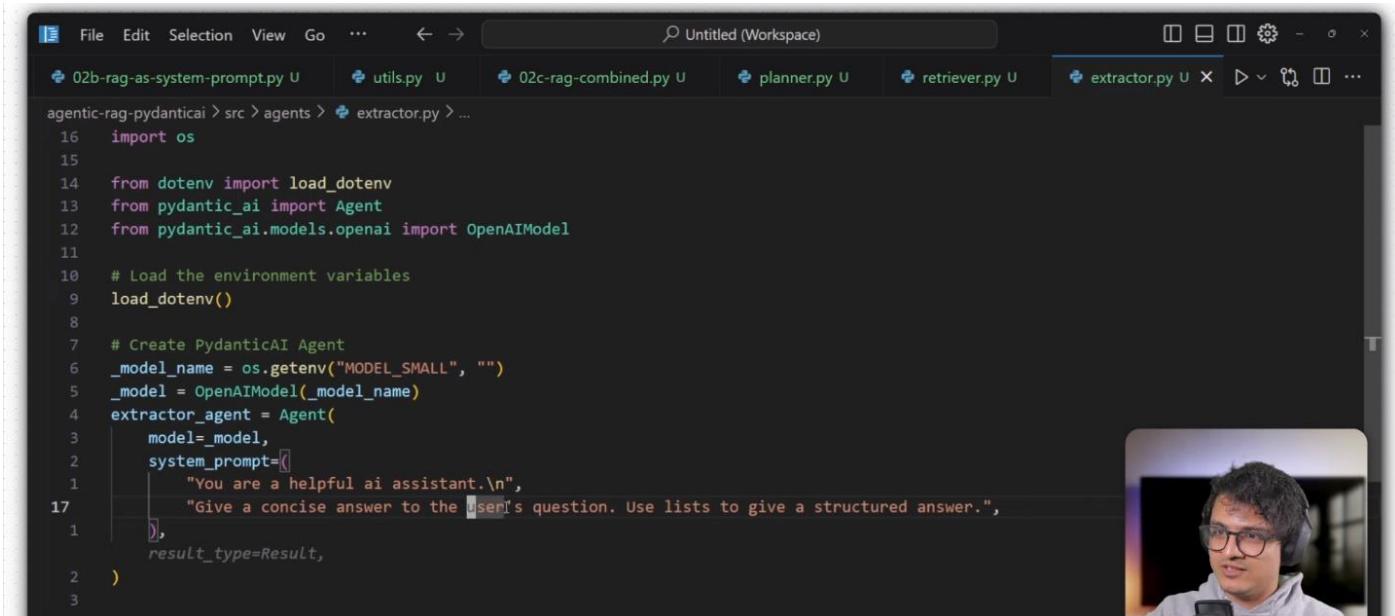
```
agentic-rag-pydanticai > src > agents > retriever.py > dynamic_system_prompt
16     model=_model,
15     system_prompt=(
14         "You are a helpful ai assistant. Help get the right information to answer the user's question. \n",
13         "Decide between using the similarity search tool or the keyword search tool based on what's the best way to search for the infor",
12         "vector search is best used for finding important which have semantic similarity to the user's question.",
11         "keyword search is best used for finding specific keywords in the database.",
10     ),
9     retries=3,
8 )
7
6
5 @retriever_agent.system_prompt
4 def dynamic_system_prompt(ctx: RunContext[None]) -> str:
3     files = read_files_as_object_array("./data")
2
1     return [
37         "Here is a list of pokemons you can search for. Make sure to include the .md extension:\n"
1         + "\n".join([f"- {file['filename']} for file in files])
2     ]
3
4
5 # Define the retrieval tool
6 @retriever_agent.tool_plain
7 def perform_similarity_search(query: str, pokemon: str) -> str:
8     """
9         Perform a similarity or vector search on the database.
10        This is best used for finding important which have semantic similarity to the user's question.
```





```
agentic-rag-pydanticai > src > agents > retriever.py > dynamic_system_prompt
7  def perform_similarity_search(query: str, pokemon: str) -> str:
17     |     pokemon = f"{pokemon}.md"
18
19     results = perform_vector_search(query, pokemon=pokemon, top_k=3)
20     return build_context_from_results(results)
21
22
23 @retriever_agent.tool_plain
24 def perform_keyword_search(keyword: str, pokemon: str) -> str:
25     """
26         Perform a keyword based full text search on the database.
27         This is best used for finding specific keywords in the database.
28         I
29     Args:
30         keyword (str): A keyword to search for in the database.
31         pokemon (str): The pokemon to search for.
32     """
33     if not pokemon.endswith(".md"):
34         pokemon = f"{pokemon}.md"
35
36     results = perform_fts_search(keyword, top_k=3, pokemon=pokemon)
37     return build_context_from_results(results)
38
```

This is our previous retriever code



```
agentic-rag-pydanticai > src > agents > extractor.py > ...
16  import os
15
14  from dotenv import load_dotenv
13  from pydantic_ai import Agent
12  from pydantic_ai.models.openai import OpenAIModel
11
10  # Load the environment variables
9   load_dotenv()
8
7  # Create PydanticAI Agent
6  _model_name = os.getenv("MODEL_SMALL", "")
5  _model = OpenAIModel(_model_name)
4  extractor_agent = Agent(
3   model=_model,
2   system_prompt=[
1    "You are a helpful ai assistant.\n",
17   "Give a concise answer to the user's question. Use lists to give a structured answer.",
1   ],
1   result_type=Result,
2 )
3
```

Then we have our extractor code

```
m-prompt.py U  utils.py U  02c-rag-combined.py U  planner.py U  retriever.py U  extractor.py U  finalizer.py U X  ▷ ▶ ⌂ ⌂ ...  
agentic-rag-pydancticai > src > agents > finalizer.py > ...  
 6 import os  
 5  
 4 from dotenv import load_dotenv  
 3 from pydantic_ai import Agent  
 2 from pydantic_ai.models.openai import OpenAIModel  
 1  
 7 # Load the environment variables  
 1 load_dotenv()  
 2  
 3 # Create PydanticAI Agent  
 4 _model_name = os.getenv("MODEL_MEDIUM", "")  
 5 _model = OpenAIModel(_model_name)  
 6 finalizer_agent = Agent(  
 7     model=_model,  
 8     system_prompt=(  
 9         "You are a helpful ai assistant.\n"  
10         "Give a descriptive & detailed answer to the user's question."  
11     ),  
12 )  
13
```





This is the deducer code



Now we can start the workflow code itself as above

File Edit Selection View Go ... ⟲ ⟳ Untitled (Workspace)

utils.py U 02c-rag-combined.py U planner.py U retriever.py U extractor.py U 03-agentic-rag.py U X agent.py D v ⚡ ...

agentic-rag-pydanticai > src > 03-agentic-rag.py > main

```
14 def main():
15     print("\n")
16
17     # Get the relevant information for each subquestion
18     contexts = []
19     for subquestion in subquestions.data.subquestions:
20         print(f"Getting relevant information for: {subquestion}")
21         retriever_result = retriever_agent.run_sync(subquestion)
22         # print(f"Retrieved relevant information:\n{retriever_result.data}")
23
24         # Extract the relevant information
25         prompt_for_extractor = (
26             f"{subquestion}\n"
27             "Carefully study the following context to answer the user's question:"
28             f"\n{retriever_result.data}"
29         )
30         extractor_result = extractor_agent.run_sync(prompt_for_extractor)
31         print(f"Extracted relevant information:\n{extractor_result.data}")
32
33         # Add the final context to the list
34         contexts.append(extractor_result.data)
35
36     # Finalize the answer
37     context = "---\n".join(contexts)
38     prompt_for_finalizer = (
39         f"\n{query}"
40         "Carefully study the following context to answer the user's question"
41         f"\n{context}"
42     )
43     extractor_result = extractor_agent.run_sync(prompt_for_finalizer)
44     print(f"Extracted relevant information:\n{extractor_result.data}")
45
46     # Add the final context to the list
47     contexts.append(extractor_result.data)
48
49     # Finalize the answer
50     context = "---\n".join(contexts)
51     prompt_for_finalizer = (
52         f"\n{query}"
53         "Carefully study the following context to answer the user's question"
54         f"\n{context}"
55     )
56     final_answer = finalizer_agent.run_sync(prompt_for_finalizer)
57     print(f"Final Answer:\n{final_answer.data}")
58
59
60 if __name__ == "__main__":
61     main()
```

WSL: Ubuntu 20.04 LTS 64-bit Python 3.10.0venv:venv



File Edit Selection View Go ... ⟲ ⟳ Untitled (Workspace)

utils.py U 02c-rag-combined.py U planner.py U retriever.py U extractor.py U 03-agentic-rag.py U X agent.py D v ⚡ ...

agentic-rag-pydanticai > src > 03-agentic-rag.py > main

```
10 def main():
11     # Get the relevant information for each subquestion
12     contexts = []
13     for subquestion in subquestions.data.subquestions:
14         print(f"Getting relevant information for: {subquestion}")
15         retriever_result = retriever_agent.run_sync(subquestion)
16         # print(f"Retrieved relevant information:\n{retriever_result.data}")
17
18         # Extract the relevant information
19         prompt_for_extractor = (
20             f"{subquestion}\n"
21             "Carefully study the following context to answer the user's question:"
22             f"\n{retriever_result.data}"
23         )
24         extractor_result = extractor_agent.run_sync(prompt_for_extractor)
25         print(f"Extracted relevant information:\n{extractor_result.data}")
26
27         # Add the final context to the list
28         contexts.append(extractor_result.data)
29
30     # Finalize the answer
31     context = "---\n".join(contexts)
32     prompt_for_finalizer = (
33         f"\n{query}"
34         "Carefully study the following context to answer the user's question"
35         f"\n{context}"
36     )
37     final_answer = finalizer_agent.run_sync(prompt_for_finalizer)
38     print(f"Final Answer:\n{final_answer.data}")
39
40
41 if __name__ == "__main__":
42     main()
```



```
> uv run src/03-agentic-rag.py
Logfire project URL: https://logfire.pydantic.dev/yourtechbud/pydantic-101
03:24:58.708 planner_agent run prompt=Between pikachu and charizard who has a more powerful normal type attack?
03:24:58.709   preparing model and tools run_step=1
03:24:58.710   model request
03:25:06.214   handle model response
Identified Subquestions:
What normal-type attack does Pikachu have?
What is the power of Pikachu's normal-type attack?
What is the highest level at which Pikachu can learn a normal-type attack?
What normal-type attack does Charizard have?
What is the power of Charizard's normal-type attack?
What is the highest level at which Charizard can learn a normal-type attack?
```



```
Getting relevant information for: What normal-type attack does Pikachu have?  
03:25:06.215 retriever_agent run prompt=What normal-type attack does Pikachu have?  
03:25:06.216 preparing model and tools run_step=1  
03:25:06.217 model request  
03:25:11.271 handle model response  
03:25:11.272 running tools=['perform_keyword_search']  
03:25:11.351 extractor_agent run prompt=What normal-type attack does Pikachu have?  
Carefully study the...le with these Hidden Machines in Pok  mon FireRed & LeafGreen:
```

```
03:25:11.353 preparing model and tools run_step=1  
03:25:11.353 model request  
03:25:13.799 handle model response
```

Extracted relevant information:

Based on the given context, here is a list of normal-type attacks that Pikachu has:

..er-compose/ai

#### **Identified Subquestions:**

What is the power of Pikachu's strongest normal type attack?

Getting relevant information for: What is the power of Pikachu's strongest normal type attack?

03:30:10.566 retriever\_agent run prompt=What is the  
03:30:10.567 reasoning model and tools run step 1

03:30:10.567 preparing mod  
03:30:10.567 model request

03:30:10.567 model request  
03:30:15.643 handle model response

03:30:15.643 handle model response  
03:30:15.645 running toolbar[Unperform keyword search!]

03:30:15.645 running tools=['perform\_keyword\_search']  
03:30:15.670 extractor agent run prompt-What is the power of Rikomatic?

...a compatible parents [can be found below] (#egg-)

03:30:15-672 preparing mod

03:30:15.672 preparing model and t  
03:30:15.672 model request

03:30:15.672 model request  
03:30:19.485 handle model res

Extracted relevant information:

Based on the given condition

+ \*\*Growl\*\*: Not applicable (configures status conditions rather than dealing damage)  
+ \*\*ThunderShock\*\*: 40 points of attack power (but it is electric-type move so it is not  
+ \*\*Tail Whip\*\*: Not applicable (configures status conditions rather than dealing damage)  
+ \*\*Thunder Wave\*\*: Not applicable (configures status conditions rather than dealing damage)  
+ \*\*Quick Attack\*\*: 40 points of attack power  
+ \*\*Double Team\*\*: Not applicable (deals no damage by itself; possibly puts the target to  
documented)



```

+ **Double Team**: Not applicable (deals no damage by itself; possibly puts the target to sleep, which is not documented)
+ **Slam**: 80 points of attack power
+ **Agility**: Not applicable (a status move with no direct attack power; as described in the Pokémon games, this move can lower an opponent's speed but that is purely mana-based)
+ **Thunder**: (Electric-type, not used for the answer)
+ **Light Screen**: Not applicable (a status move with no direct attack power).

The most powerful Normal-type attack available and learned by Pikachu in Pokémon Ruby & Sapphire and Pokémon FiReRed & LeafGreen at the given levels is **Slam** which is 80 points of attack power, followed by **Quick Attack** which is also 40 points of attack power.
Getting relevant information for: What is the power of Charizard's strongest normal type attack?
03:30:19.488 retriever_agent run prompt=What is the power of Charizard's strongest normal type attack?
03:30:19.489 preparing model and tools run_step=1
03:30:19.490 model request
03:30:24.442 handle model response
03:30:24.443 running tools=['perform_keyword_search']
03:30:24.452 extractor_agent run prompt=What is the power of Charizard's strongest normal type attack? [can be found below] (#egg-move-parents).

03:30:24.454 preparing model and tools run_step=1
03:30:24.454 model request
03:30:26.087 handle model response
Extracted relevant information:
**Strongest Normal Type Attack of Charizard:***
- Charizard's strongest normal type attack is **Slash**.
- Slash's power is **70**, which is the highest power among Charizard's normal type attacks.
03:30:26.090 finalizer_agent run prompt=Between pikachu and charizard who has a more powerful normal type attack in Pokémon.

```



```

03:25:58.935 model request
03:26:10.106 handle model response
Final Answer:
To answer the user's question regarding which of Pikachu or Charizard has a more powerful normal-type attack, I'll compare the information provided for the normal-type moves of each Pokémon.

**Pikachu:***
- Normal-type attacks listed include: Body Slam (Power: 85), Double-Edge (Power: 120), Endure (no damage), Mega Kick (Power: 120), Mega Punch (Power: 80), Mimic (no damage), Substitute (no damage), Bide (no specified power), and Slam (Power: 80).
- The most powerful normal-type attack Pikachu can learn is **Mega Kick** or **Double-Edge**, both with a power of **120**.

**Charizard:***
- Normal-type attacks listed include: Body Slam (Power: 85), Double-Edge (Power: 120), Endure (no damage), Mega Kick (Power: 120), Mega Punch (Power: 80), Mimic (no damage), Sleep Talk (no damage), Snore (Power: 40), Substitute (no damage), Swords Dance (no damage).
- The most powerful normal-type attack Charizard can learn is **Mega Kick** or **Double-Edge**, both with a power of **120**.

**Conclusion:***
Both Pikachu and Charizard have the same most powerful normal-type attack, which is **Double-Edge**, both with a power of **120**. Therefore, they have equally powerful normal-type attacks.
>
>

~/work/ytb-codes/ytb-practical-guide/agentic-rag-pydanticai ft/agentic-rag-pydanticai *1

```



<https://pokemondb.net/pokedex/pikachu/moves/3>

Pikachu learns the following moves in Pokémon Ruby & Sapphire at the levels specified.

Lv.	Move	Type	Cat.	Power	Acc.
1	Growl	NORMAL	●	—	100
1	ThunderShock	ELECTRIC	●	40	100
6	Tail Whip	NORMAL	●	—	100
8	Thunder Wave	ELECTRIC	●	—	100
11	Quick Attack	NORMAL	★	40	100
15	Double Team	NORMAL	●	—	—
20	Slam	NORMAL	★	80	75
26	Thunderbolt	ELECTRIC	●	95	100
33	Agility	PSYCHIC	●	—	—
41	Thunder	ELECTRIC	●	120	70
50	Light Screen	PSYCHIC	●	—	—

**Egg moves**

Pikachu learns the following moves via breeding in Pokémon Ruby & Sapphire. Details and compatible parents can be found below.

Move	Type	Cat.	Power	Acc.
Bide	NORMAL	★	—	100
Charge	ELECTRIC	●	—	—
DoubleSlap	NORMAL	★	15	85
Encore	NORMAL	●	—	100
Present	NORMAL	★	—	90
Reversal	FIGHTING	★	—	100

**Moves learnt by HM**

Pikachu is compatible with these Hidden Machines in Pokémon Ruby & Sapphire:

HM	Move	Type	Cat.	Power	Acc.
04	Strength	NORMAL	★	80	100
05	Flash	NORMAL	●	—	70
06	Rock Smash	FIGHTING	★	20	100

**Moves learnt by TM**

Pikachu is compatible with these Technical Machines in Pokémon Ruby & Sapphire:

TM	Move	Type	Cat.	Power	Acc.
01	Focus Punch	FIGHTING	★	150	100
06	Toxic	POISON	●	—	85
10	Hidden Power	NORMAL	★	60	100
16	Light Screen	PSYCHIC	●	—	—
17	Protect	NORMAL	●	—	—
18	Rain Dance	WATER	●	—	—
21	Frustration	NORMAL	★	—	100
23	Iron Tail	STEEL	★	100	75
24	Thunderbolt	ELECTRIC	●	95	100
25	Thunder	ELECTRIC	●	120	70
27	Return	NORMAL	★	—	100
28	Dig	GROUND	★	60	100
31	Brick Break	FIGHTING	★	75	100
32	Double Team	NORMAL	●	—	—
34	Shock Wave	ELECTRIC	●	60	∞



<https://pokemondb.net/pokedex/charizard/moves/3>

Charizard learns the following moves in Pokémon Ruby & Sapphire at the levels specified.

Lv.	Move	Type	Cat.	Power	Acc.
1	Ember	FIRE	●	40	100
1	Growl	NORMAL	●	—	100
1	Scratch	NORMAL	★	40	100
1	SmokeScreen	NORMAL	●	—	100
7	Ember	FIRE	●	40	100
13	SmokeScreen	NORMAL	●	—	100
20	Rage	NORMAL	★	20	100
27	Scary Face	NORMAL	●	—	90
34	Flamethrower	FIRE	●	95	100
36	Wing Attack	FLYING	★	60	100
44	Slash	NORMAL	●	70	100
54	Dragon Rage	DRAGON	●	—	100
64	Fire Spin	FIRE	●	15	70

**Egg moves**

Charizard learns the following moves via breeding in Pokémon Ruby & Sapphire. Details and compatible parents can be found below.

Move	Type	Cat.	Power	Acc.
------	------	------	-------	------

**Moves learnt by HM**

Charizard is compatible with these Hidden Machines in Pokémon Ruby & Sapphire:

HM	Move	Type	Cat.	Power	Acc.
01	Cut	NORMAL	★	50	95
02	Fly	FLYING	★	70	95
04	Strength	NORMAL	★	80	100
06	Rock Smash	FIGHTING	★	20	100

**Moves learnt by TM**

Charizard is compatible with these Technical Machines in Pokémon Ruby & Sapphire:

TM	Move	Type	Cat.	Power	Acc.
01	Focus Punch	FIGHTING	★	150	100
02	Dragon Claw	DRAGON	●	80	100
05	Roar	NORMAL	●	—	100
06	Toxic	POISON	●	—	85
10	Hidden Power	NORMAL	★	60	100
11	Sunny Day	FIRE	●	—	—
15	Hyper Beam	NORMAL	★	150	90
17	Protect	NORMAL	●	—	—
21	Frustration	NORMAL	★	—	100
23	Iron Tail	STEEL	★	100	75



```
03:30:26.090 finalizer_agent run prompt=Between pikachu and charizard who has a more powerful normal t...ighest power among Charizard's normal type attacks in Pok  mon.
03:30:26.090 preparing model and tools run_step=1
03:30:26.090 model request
03:30:33.093 handle model response
Final Answer:
To determine which Pok  mon has a more powerful Normal-type attack between Pikachu and Charizard, we need to compare their strongest Normal-type moves:

1. **Pikachu's Normal-type Attacks**:
   - **Quick Attack**: 40 points of attack power.
   - **Slam**: 80 points of attack power.

   Therefore, Pikachu's most powerful Normal-type attack is **Slam**, with an attack power of **80**.

2. **Charizard's Normal-type Attacks**:
   - **Slash**: 70 points of attack power.

   Therefore, Charizard's most powerful Normal-type attack is **Slash**, with an attack power of **70**.

Comparing these two moves:
- **Pikachu's Slam**: 80 points of attack power.
- **Charizard's Slash**: 70 points of attack power.

Hence, **Pikachu** has a more powerful Normal-type attack with **Slam**, which has 80 points of attack power compared to Charizard's **Slash**, which has 70 points of attack power.
```







A screenshot of a code editor showing Python code for an AI agent. The code defines an Agent named 'retriever\_agent' with a system prompt about being a helpful AI assistant. It also includes a dynamic\_system\_prompt function that lists files from a directory and a perform\_similarity\_search function that performs a similarity search on a database.

```
File Edit Selection View Go ... ← → ⌂ Untitled (Workspace) ⌂ ⌂ ⌂ ⌂ ⌂ ⌂ ⌂ ...  
utils.py U 02c-rag-combined.py U planner.py U retriever.py X extractor.py U 03-agentic-rag.py U agent.py D v ⌂ ...  
agentic-rag-pydanica1ai > src > agents > retriever.py > retriever_agent  
20 retriever_agent = Agent(  
21     model=_model,  
22     system_prompt=  
23         "You are a helpful ai assistant. Help get the right information to answer the user's question. \n",  
24         "Decide between using the similarity search tool or the keyword search tool based on what's the best way to search for the infor  
25         "vector search is best used for finding important which have semantic similarity to the user's question.",  
26         "keyword search is best used for finding specific keywords in the database.",  
27     ),  
28     retries=3,  
29 )  
30  
31  
32 @retriever_agent.system_prompt  
33 def dynamic_system_prompt(ctx: RunContext[None]) -> str:  
34     files = read_files_as_object_array("./data")  
35  
36     return (  
37         "Here is a list of pokemons you can search for. Make sure to include the .md extension:\n"  
38         + "\n".join([f"- {file['filename']} for file in files])  
39     )  
40  
41  
42 # Define the retrieval tool  
43 @retriever_agent.tool_plain  
44 def perform_similarity_search(query: str, pokemon: str) -> str:  
45     """  
46         Perform a similarity or vector search on the database.  
47     """
```

```
utils.py U 02c-rag-combined.py U planner.py U retriever.py X extractor.py U 03-agentic-rag.py U agent.py D v ⌂ ...
```

agentic-rag-pydantical > src > agents > retriever.py > perform\_keyword\_search

```
36
35     # @retriever_agent.system_prompt
34     # def dynamic_system_prompt(ctx: RunContext[None]) -> str:
33     #     files = read_files_as_object_array("./data")
32
31     #     return (
30     #         "Here is a list of pokemons you can search for. Make sure to include the .md extension:\n"
29     #         + "\n".join([f"- {file['filename']}\" for file in files])
28     #     )
27
26
25     # Define the retrieval tool
24     @retriever_agent.tool_plain
23     def perform_similarity_search(query: str) -> str:
22         """
21             I
20             Perform a similarity or vector search on the database.
21             This is best used for finding important which have semantic similarity to the user's question.
19
18             Args:
17                 query (str): A concise question for which you need to get the most relevant information.
16
15
14             results = perform_vector_search(query, top_k=3)
13             return build_context_from_results(results)
12
11
```

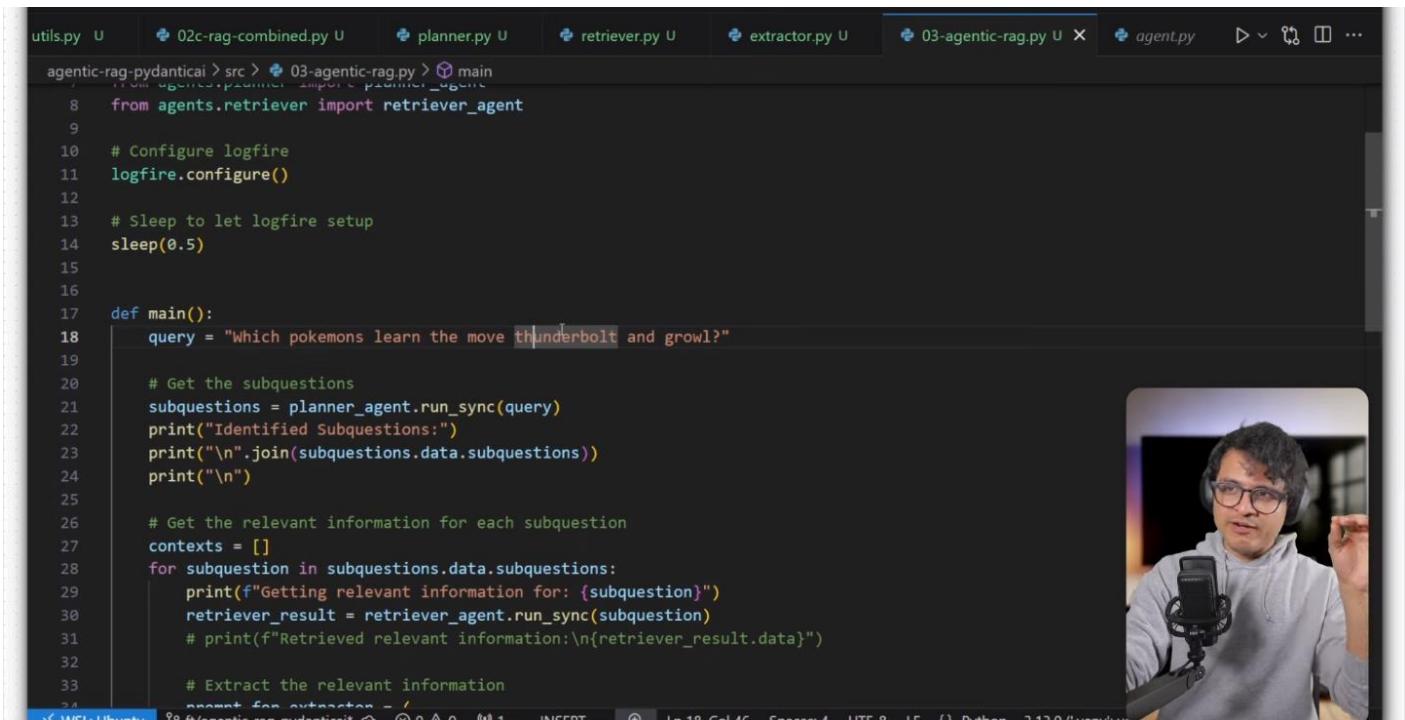


```
utils.py U 02c-rag-combined.py U planner.py U retriever.py U extractor.py U 03-agentic-rag.py U agent.py D v ⌂ ...
```

agentic-rag-pydantical > src > agents > retriever.py > perform\_keyword\_search

```
23     def perform_similarity_search(query: str) -> str:
20         This is best used for finding important which have semantic similarity to the user's question.
19
18             Args:
17                 query (str): A concise question for which you need to get the most relevant information.
16
15
14             results = perform_vector_search(query, top_k=3)
13             return build_context_from_results(results)
12
11
10     @retriever_agent.tool_plain
9     def perform_keyword_search(keyword: str, pokemon: str) -> str:
8         """
7             Perform a keyword based full text search on the database.
6             This is best used for finding specific keywords in the database.
5
4             Args:
3                 keyword (str): A keyword to search for in the database.
2
1
66             results = perform_fts_search(keyword, top_k=3)
1             return build_context_from_results(results)
2
```





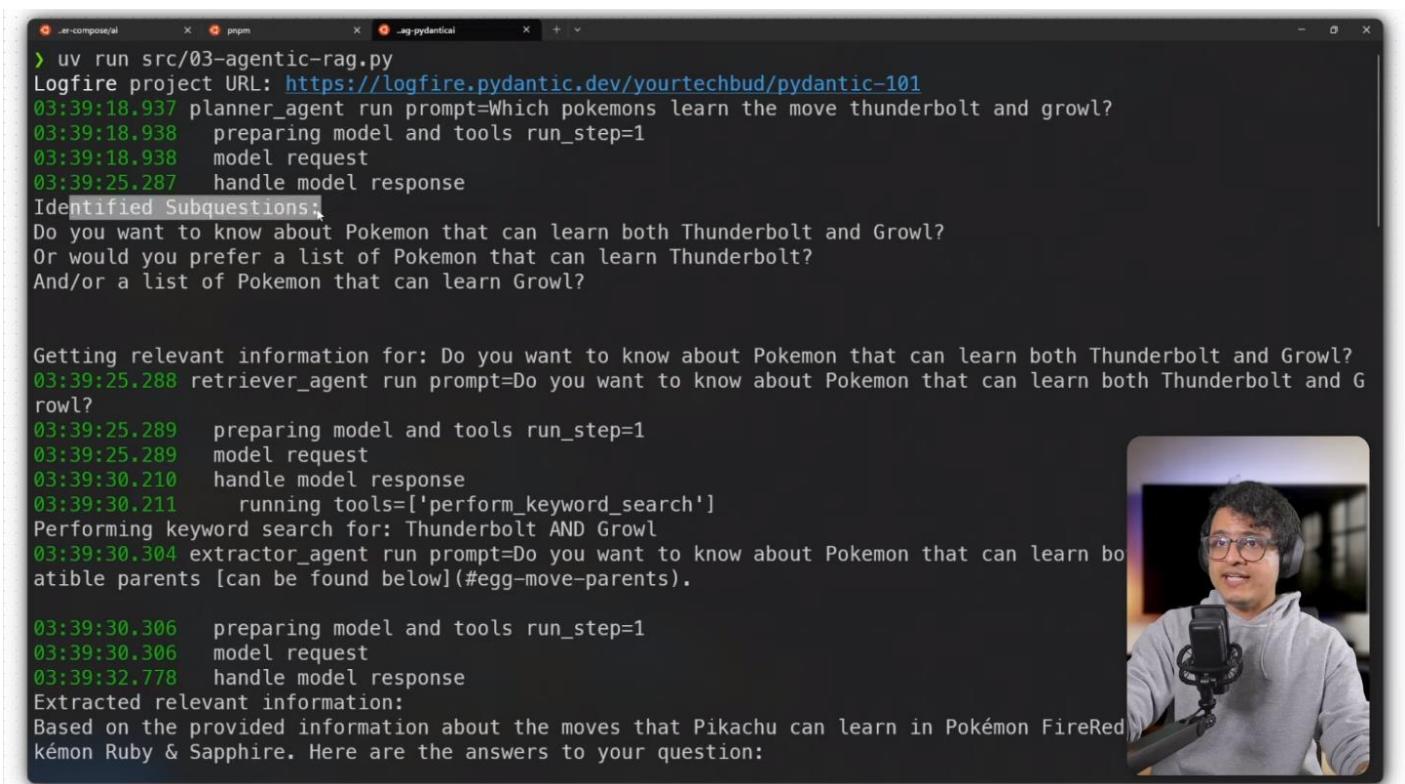
```

utils.py U 02c-rag-combined.py U planner.py U retriever.py U extractor.py U 03-agentic-rag.py U agent.py D v ...
agentic-rag-pydantic > src > 03-agentic-rag.py > main
8     from agents.retriever import retriever_agent
9
10    # Configure logfire
11    logfire.configure()
12
13    # Sleep to let logfire setup
14    sleep(0.5)
15
16
17 def main():
18     query = "Which pokemons learn the move thunderbolt and growl?"
19
20     # Get the subquestions
21     subquestions = planner_agent.run_sync(query)
22     print("Identified Subquestions:")
23     print("\n".join(subquestions.data.subquestions))
24     print("\n")
25
26     # Get the relevant information for each subquestion
27     contexts = []
28     for subquestion in subquestions.data.subquestions:
29         print(f"Getting relevant information for: {subquestion}")
30         retriever_result = retriever_agent.run_sync(subquestion)
31         # print(f"Retrieved relevant information:\n{retriever_result.data}")
32
33         # Extract the relevant information
34         # comment for extraction = ...

```



This will require scans across all pokemons



```

uv run src/03-agentic-rag.py
Logfire project URL: https://logfire.pydantic.dev/yourtechbud/pydantic-101
03:39:18.937 planner_agent run prompt=Which pokemons learn the move thunderbolt and growl?
03:39:18.938 preparing model and tools run_step=1
03:39:18.938 model request
03:39:25.287 handle model response
Identified Subquestions:
Do you want to know about Pokemon that can learn both Thunderbolt and Growl?
Or would you prefer a list of Pokemon that can learn Thunderbolt?
And/or a list of Pokemon that can learn Growl?

Getting relevant information for: Do you want to know about Pokemon that can learn both Thunderbolt and Growl?
03:39:25.288 retriever_agent run prompt=Do you want to know about Pokemon that can learn both Thunderbolt and Growl?
03:39:25.289 preparing model and tools run_step=1
03:39:25.289 model request
03:39:30.210 handle model response
03:39:30.211 running tools=['perform_keyword_search']
Performing keyword search for: Thunderbolt AND Growl
03:39:30.304 extractor_agent run prompt=Do you want to know about Pokemon that can learn both Thunderbolt and Growl?
03:39:30.305 preparing model and tools run_step=1
03:39:30.305 model request
03:39:32.778 handle model response
Extracted relevant information:
Based on the provided information about the moves that Pikachu can learn in Pok  mon FireRed and Pok  mon Ruby & Sapphire. Here are the answers to your question:

```



```
atable parents [can be found below](#egg-move-parents).  
03:39:30.306 preparing model and tools run_step=1  
03:39:30.306 model request  
03:39:32.778 handle model response  
Extracted relevant information:  
Based on the provided information about the moves that Pikachu can learn in Pok  mon FireRed & LeafGreen, and Pok  mon Ruby & Sapphire. Here are the answers to your question:  
  
**Pok  mon that can learn both Thunderbolt and Growl in Pok  mon FireRed & LeafGreen:**  
  
* Pikachu  
  
**Reason:** According to the information provided, Pikachu learns Thunderbolt at level 26 and Growl at level 1 in Pok  mon FireRed & LeafGreen.  
  
**Pok  mon that can learn both Thunderbolt and Growl in Pok  mon Ruby & Sapphire:**  
  
* Pikachu  
  
**Reason:** According to the information provided, Pikachu learns Thunderbolt at level 26 and Growl at level 1 in Pok  mon Ruby & Sapphire.  
Getting relevant information for: Or would you prefer a list of Pokemon that can learn Thunderbolt?  
03:39:32.781 retriever_agent run prompt=Or would you prefer a list of Pokemon that can learn Thunderbolt?  
03:39:32.782 preparing model and tools run_step=1  
03:39:32.782 model request  
03:39:37.217 handle model response  
03:39:37.218 running tools=['perform_keyword_search']  
Performing keyword search for: Thunderbolt
```



```
**Pok  mon that can learn both Thunderbolt and Growl in Pok  mon FireRed & LeafGreen:**  
  
* Pikachu  
  
**Reason:** According to the information provided, Pikachu learns Thunderbolt at level 26 and Growl at level 1 in Pok  mon FireRed & LeafGreen.  
  
**Pok  mon that can learn both Thunderbolt and Growl in Pok  mon Ruby & Sapphire:**  
  
* Pikachu  
  
**Reason:** According to the information provided, Pikachu learns Thunderbolt at level 26 and Growl at level 1 in Pok  mon Ruby & Sapphire.  
Getting relevant information for: Or would you prefer a list of Pokemon that can learn Thunderbolt?  
03:39:32.781 retriever_agent run prompt=Or would you prefer a list of Pokemon that can learn Thunderbolt?  
03:39:32.782 preparing model and tools run_step=1  
03:39:32.782 model request  
03:39:37.217 handle model response  
03:39:37.218 running tools=['perform_keyword_search']  
Performing keyword search for: Thunderbolt  
03:39:37.228 extractor_agent run prompt=Or would you prefer a list of Pokemon that can learn Thunderbolt? tutors ([details](/mechanics/move-tutors#tutor-frlg)):  
  
03:39:37.230 preparing model and tools run_step=1  
03:39:37.230 model request  
03:39:39.424 handle model response  
Extracted relevant information:  
Based on the provided information, here is a list of Pok  mon that can learn Thunderbolt:
```



```
03:39:37.218     running tools=['perform_keyword_search']
Performing keyword search for: Thunderbolt
03:39:37.228 extractor_agent run prompt=Or would you prefer a list of Pokemon that can learn Thunderbo...m move
tutors ([details](/mechanics/move-tutors#tutor-frlg)):

03:39:37.230     preparing model and tools run_step=1
03:39:37.230     model request
03:39:39.424     handle model response
Extracted relevant information:
Based on the provided information, here is a list of Pok  mon that can learn Thunderbolt:

1. **Pikachu**:
   - In Pok  mon Ruby & Sapphire: Learns Thunderbolt at level 26.
   - In Pok  mon FireRed & LeafGreen: Learns Thunderbolt at level 1.
   - In Pok  mon Emerald: Learns Thunderbolt at level 1.

2. **Raichu**:
   - In Pok  mon FireRed & LeafGreen: Learns Thunderbolt at level 1.
   - In Pok  mon Emerald: Learns Thunderbolt at level 1.

Getting relevant information for: And/or a list of Pokemon that can learn Growl?
03:39:39.427 retriever_agent run prompt=And/or a list of Pokemon that can learn Growl?
03:39:39.427     preparing model and tools run_step=1
03:39:39.427     model request
03:39:43.452     handle model response
03:39:43.453     running tools=['perform_keyword_search']

Performing keyword search for: Growl
03:39:43.459 extractor_agent run prompt=And/or a list of Pokemon that can learn Growl?
Carefully study...d compatible parents [can be found below](#egg-move-parents).
```



```
2. **Raichu**:
   - In Pok  mon FireRed & LeafGreen: Learns Thunderbolt at level 1.
   - In Pok  mon Emerald: Learns Thunderbolt at level 1.

Getting relevant information for: And/or a list of Pokemon that can learn Growl?
03:39:39.427 retriever_agent run prompt=And/or a list of Pokemon that can learn Growl?
03:39:39.427     preparing model and tools run_step=1
03:39:39.427     model request
03:39:43.452     handle model response
03:39:43.453     running tools=['perform_keyword_search']

Performing keyword search for: Growl
03:39:43.459 extractor_agent run prompt=And/or a list of Pokemon that can learn Growl?
Carefully study...d compatible parents [can be found below](#egg-move-parents).

03:39:43.460     preparing model and tools run_step=1
03:39:43.461     model request
03:39:45.320     handle model response
Extracted relevant information:
Based on the provided information, the Pok  mon species that can learn Growl are:

1. **Slowpoke**:
   - Slowpoke can learn Growl at level 6 in Pok  mon Ruby & Sapphire, Pok  mon Emerald, and Pok  mon FireGreen.

Please note that other Pok  mon species might also learn Growl through other methods, such as leveling up. However, the provided information only mentions the level-up method for Slowpoke.
03:39:45.322 finalizer_agent run prompt=Which pokemons learn the move thunderbolt and growl?
03:39:45.322     preparing model and tools run_step=1
```



```
03:39:45.322 finalizer_agent run prompt=Which pokemons learn the move thunderbolt and growl?Carefully ...ed information only mentions the level-up method for Slowpoke.  
03:39:45.323 preparing model and tools run_step=1  
03:39:45.323 model request  
03:39:53.010 handle model response  
Final Answer:  
Based on the provided information, the following Pokémon can learn both Thunderbolt and Growl:  
  
**Pikachu:**  
- **Thunderbolt:**  
- *Pokémon Ruby & Sapphire:* Learns at level 26.  
- *Pokémon FireRed & LeafGreen:* Earns at level 1.  
- *Pokémon Emerald:* Learns at level 1.  
- **Growl:**  
- *Pokémon Ruby & Sapphire:* Learns at level 1.  
- *Pokémon FireRed & LeafGreen:* Learns at level 1.  
- *Pokémon Emerald:* Earning details for Growl in Emerald are not provided, but it is inferred its level-up moveset similar to the other games.
```

So, Pikachu in both Pokémon Ruby & Sapphire and Pokémon FireRed & LeafGreen can learn both Thunderbolt and Growl. The specifics for Pokémon Emerald suggest Pikachu has a well-rounded electric-type move set that includes both Thunderbolt and Growl from early levels.

To summarize, Pikachu is the Pokémon consistently mentioned in the provided information to learn both Thunderbolt and Growl. Raichu and Slowpoke were mentioned for specific moves, but not both moves together. The information for Pikachu covers both moves from early levels.

```
~$ /work/ytb-codes/ytb-practical-guide/agentic-rag-pydanticai ft/agentic-rag-pydanticai *1
```

