anc_ollama

February 6, 2026

0.1 A Geology Flaneur needs an Agentic Companion

Imagine a traveler wearing **Apple Vision Pro**, moving through mountain corridors as a true **geology flaneur**: observing roadcuts, folds, and landforms in context, with grounded explanations layered into the moment.

This notebook is the proof path toward that experience: route-aware field guidance, citation-backed answers, and quality-gated reasoning that can later power an immersive spatial interface.

1 ANC: Wikipedia with (FAISS, Ollama and StateGraph)

This notebook combines two Ollama workflows in one place:

- Baseline notebook pipeline: Wikipedia -> chunking -> FAISS -> local Q&A with `ChatOllama`
- Canonical orchestration diagram + run flow via shared `naturalist_companion.stategraph_shared`

1.1 POC Motivation and Evidence

1.1.1 Why this notebook exists

This notebook is the local Ollama proof-of-concept for route-grounded geology guidance. It validates grounded retrieval + generation + quality checks before full app productionization.

1.1.2 Hypotheses under test

1. Local Ollama embeddings and LLM models can produce useful grounded answers.
2. Wikipedia-only retrieval can support safety-aware field guidance with citations.
3. Shared orchestration (`stategraph_shared`) can enforce quality gates and stable output structure.

1.1.3 How to capture comparable evidence

- Record Stage 1/2/3 and Stage 3b timings from notebook logs.
- Run the StateGraph eval harness and capture pass rate + citation validity.
- Keep model names/config values with each run artifact for apples-to-apples comparisons.

1.1.4 What this run does not prove yet

- Multi-user serving scale and concurrency behavior
- Persistent shared retrieval indexes for app traffic

- End-to-end mobile app integration with route + camera UX
- Production SLO/cost governance

1.2 Prerequisites (Run First)

1.2.1 Local Ollama setup

- Ensure Ollama is installed and running: `ollama serve`
- Pull the embedding model: `ollama pull nomic-embed-text`
- Pull the chat model used in this notebook: `ollama pull llama3.1:8b`
- Optional smaller/faster local model: `ollama pull llama3.2:3b`

1.2.2 Optional notebook installs

Run the install cell below only if preflight reports missing packages.

[1]:
```
# Uncomment in fresh environments or if preflight fails:

%pip install -q -U -r ../requirements-ollama-dev.txt

# Optional only if rich widgets are missing:
# %pip install -q -U ipywidgets jupyterlab_widgets
```

```
Note: you may need to restart the kernel to use updated packages.
```

[2]:
```
import importlib
import json
import re
import os
import sys
from pathlib import Path
from typing import Literal
import warnings
from threading import Event, Thread
from IPython.display import Image, Markdown, display
from time import perf_counter
```

1.2.3 Dependency Preflight (Fail Fast)

[3]:
```
def _candidate_src_paths():
    candidates = []

    # Optional explicit override.
    env_src = os.environ.get("NATURALIST_COMPANION_SRC", "").strip()
    if env_src:
        candidates.append(Path(env_src))

    cwd = Path.cwd()
    candidates.extend([
```

```
        cwd / "src",
        cwd.parent / "src",
        cwd.parent.parent / "src",
    ])

    # Optional scan for repository-style checkouts.
    repos_root = Path("/Workspace/Repos")
    if repos_root.exists():
        for pkg_dir in repos_root.glob("*/*/src/naturalist_companion"):
            candidates.append(pkg_dir.parent)

    deduped = []
    seen = set()
    for item in candidates:
        key = str(item)
        if key in seen:
            continue
        seen.add(key)
        deduped.append(item)
    return deduped


for src_path in _candidate_src_paths():
    if (src_path / "naturalist_companion").exists() and str(src_path) not in␣
 ↪sys.path:
        sys.path.insert(0, str(src_path))
        break


CHECKS = [
    ("FAISS backend", ["faiss"]),
    ("LangGraph runtime", ["langgraph"]),
    ("Wikipedia loader module", ["langchain_community.document_loaders",␣
 ↪"langchain.document_loaders"]),
    ("Text splitter module", ["langchain_text_splitters", "langchain.
 ↪text_splitter"]),
    ("LangChain vectorstore module", ["langchain_community.vectorstores"]),
    ("LangChain in-memory docstore module", ["langchain_community.docstore.
 ↪in_memory"]),
    ("Ollama integration", ["langchain_ollama"]),
    ("Naturalist stategraph module", ["naturalist_companion.
 ↪stategraph_shared"]),
]
resolved = {}
missing = []

for label, module_candidates in CHECKS:
```

```
    matched = None
    last_error = None
    for module_name in module_candidates:
        try:
            importlib.import_module(module_name)
            matched = module_name
            break
        except ImportError as import_error:
            last_error = f"{type(import_error).__name__}: {import_error}"

    if matched is not None:
        resolved[label] = matched
    else:
        missing.append((label, module_candidates, last_error))

if missing:
    non_stategraph_missing = [m for m in missing if m[0] != "Naturalist␣
 ↪stategraph module"]
    if non_stategraph_missing:
        missing = non_stategraph_missing

    lines = ["[preflight] Missing required notebook dependencies:"]
    for label, module_candidates, last_error in missing:
        lines.append(f"- {label}: expected one of {', '.
 ↪join(module_candidates)}")
        if last_error:
            lines.append(f"  last error: {last_error}")

    lines.append("")
    lines.append("Run the install cell above, restart the kernel, and retry.")
    lines.append("Install hint: %pip install -q -U -r ../
 ↪requirements-ollama-dev.txt")
    lines.append("Set NATURALIST_COMPANION_SRC to your repo src path if needed.
 ↪")
    raise ModuleNotFoundError("\n".join(lines))

print("[preflight] Dependency check passed.")
for label, module_name in resolved.items():
    print(f"  - {label}: {module_name}")
```

```
[preflight] Dependency check passed.
  - FAISS backend: faiss
  - LangGraph runtime: langgraph
  - Wikipedia loader module: langchain_community.document_loaders
  - Text splitter module: langchain_text_splitters
  - LangChain vectorstore module: langchain_community.vectorstores
  - LangChain in-memory docstore module: langchain_community.docstore.in_memory
```

- Ollama integration: langchain_ollama
- Naturalist stategraph module: naturalist_companion.stategraph_shared

1.2.4 Notebook Imports + Runtime Config

[4]:
```
# Mitigate common macOS OpenMP duplicate-library crashes in notebook kernels.
os.environ.setdefault("KMP_DUPLICATE_LIB_OK", "TRUE")

# Silence noisy tqdm widget warning in IDE notebooks when rich progress widgets␣
 ↪are unavailable.
warnings.filterwarnings("ignore", message=".*IProgress not found.*")


# LangChain moved WikipediaLoader in newer releases; keep backward␣
 ↪compatibility.
try:
    from langchain_community.document_loaders import WikipediaLoader
except ImportError:
    from langchain.document_loaders import WikipediaLoader

try:
    from langchain_text_splitters import RecursiveCharacterTextSplitter
except ImportError:
    from langchain.text_splitter import RecursiveCharacterTextSplitter

from langchain_community.vectorstores import FAISS
from langchain_ollama import ChatOllama, OllamaEmbeddings


def _start_heartbeat(task_label: str, every_s: float = 8.0):
    stop = Event()

    def _run():
        elapsed = 0.0
        while not stop.wait(every_s):
            elapsed += every_s
            print(f"[{task_label}] still running... {elapsed:.0f}s elapsed")

    thread = Thread(target=_run, daemon=True)
    thread.start()
    return stop


from naturalist_companion.wikipedia_tools import␣
 ↪display_openstreetmap_for_pages, display_wikipedia_images_for_pages

try:
```

```
    from langchain_core.documents import Document
except ImportError:
    from langchain.schema import Document

from naturalist_companion.fallback_data import fallback_wiki_pages


STATEGRAPH_AVAILABLE = False
_stategraph_import_error = None
try:
    from naturalist_companion.stategraph_shared import (
        build_stategraph_app,
        run_i81_eval_harness,
        run_stategraph,
    )
    STATEGRAPH_AVAILABLE = True
except Exception as stategraph_import_error:
    _stategraph_import_error = stategraph_import_error
    print(f"[stategraph] import error: {type(stategraph_import_error).__name__}:
 ↪ {stategraph_import_error}")
    print("[stategraph] StateGraph cells can be skipped until this import works.
 ↪")


def _show_ollama_status() -> None:
    base_url = os.environ.get("OLLAMA_BASE_URL", "http://localhost:11434").
 ↪strip()
    print(f"[env] Ollama base URL: {base_url}")
    print("[env] Verify local models with: `ollama list`")


def _raise_ollama_hint(stage: str, model: str, base_url: str, error: Exception)␣
 ↪-> None:
    message = str(error)
    hints = [
        f"[{stage}] Ollama call failed for model={model!r} at␣
 ↪base_url={base_url!r}.",
        f"Underlying error: {message}",
    ]

    lower_msg = message.lower()
    if "connection" in lower_msg or "refused" in lower_msg or "timed out" in␣
 ↪lower_msg:
        hints.append("Ensure Ollama is running and reachable: `ollama serve`.")
    if "not found" in lower_msg or "pull" in lower_msg:
        hints.append(f"Ensure the model exists locally: `ollama pull {model}`.")
```

```
    raise RuntimeError("\n".join(hints)) from error


_show_ollama_status()
```

```
[env] Ollama base URL: http://localhost:11434
[env] Verify local models with: `ollama list`
```

1.2.5 Config (Define LLMs, Embeddings, Vector Store, Data Loader specs)

[5]:
```
# DataLoader Config
query_terms = [
    "roadcut",
    "geology",
    "sedimentary rock",
    "stratigraphy",
]
max_docs = 12  # Realistic retrieval setting for richer context.

# Stage 2 chunking + batching controls (keep small for interactive runs).
chunk_size = 1200
chunk_overlap = 150
embedding_batch_size = 8


# Retriever Config
k = 4
EMBEDDING_MODEL = os.environ.get("OLLAMA_EMBEDDING_MODEL", "nomic-embed-text")
OLLAMA_BASE_URL = os.environ.get("OLLAMA_BASE_URL", "http://localhost:11434")


# LLM Config
LLM_MODEL = os.environ.get("OLLAMA_LLM_MODEL", "llama3.1:8b")
TEMPERATURE = 0.25


# Response style controls (Roadside Geology audience: curious drivers,␣
 ↪practical field learners).
RESPONSE_TONE = "field-guide"
MAX_BULLETS_PER_SECTION = 4


# Local artifact settings
FAISS_NAMESPACE = "anc_ollama"
```

```
# StateGraph Config
STATEGRAPH_PROVIDER: Literal["ollama"] = "ollama"
STATEGRAPH_LIVE_MAX_DOCS = 16
STATEGRAPH_COMMON_CONFIG = {
    "max_retrieval_attempts": 3,
    "citation_coverage_threshold": 0.80,
    "runtime_mode": "realistic",
    "llm_temperature": TEMPERATURE,
    "llm_model": LLM_MODEL,
    "ollama_base_url": OLLAMA_BASE_URL,
    "live_max_docs": STATEGRAPH_LIVE_MAX_DOCS,
}
STATEGRAPH_RUN_CONFIG = {"artifact_root": "out/stategraph/notebook_runs",␣
 ↪**STATEGRAPH_COMMON_CONFIG}
STATEGRAPH_EVAL_CONFIG = {"artifact_root": "out/stategraph/notebook_eval",␣
 ↪**STATEGRAPH_COMMON_CONFIG}
```

1.3 Query Prompt (Edit This Cell)

Use the next code cell to set the active question(s).

Question types this notebook is designed for: - Detour geology: legal pull-offs or short walks near a route segment - Safety-first prompts: where to stop and what to avoid roadside - Route constraints: city/exit anchors plus max detour minutes - Beginner field interpretation: what visual clues to look for and why they matter

Tip: Include your nearest city or exit and your max detour time to improve stop recommendations.

[6]:
```
example_question = "I am on I-81 near Hagerstown with a 30-minute detour. Where␣
 ↪can I safely stop to observe folded Valley-and-Ridge strata, and what␣
 ↪exactly should I look for?"

example_questions = [
    "I am driving I-81 near Bristol, TN. Give me two legal pull-off stops where␣
 ↪I can see clear sedimentary layering, and tell me exactly what to look for.",
    "Near I-81 between Winchester and Strasbourg, where can I safely stop to␣
 ↪see Valley-and-Ridge structure, and what field clues confirm folding?",
    "I have 45 minutes near Hagerstown, MD. What roadside geology stop gives␣
 ↪the best payoff for a beginner, and what story does the outcrop tell?",
    "Along I-81 in the Shenandoah Valley, point me to a short-walk stop to␣
 ↪compare rock type and landform, then explain why that match matters.",
    "On an I-81 drive day, suggest one stop where I can observe evidence of␣
 ↪ancient seas or sediment transport, with specific visual clues.",
]

# StateGraph run can use the same prompt by default; edit independently if␣
 ↪desired.
stategraph_question = example_question
```

```python
def _parse_place_query_list(raw: str, max_items: int = 6) -> list[str]:
    text = str(raw or "").strip()
    if not text:
        return []
    match = re.search(r"\[[\s\S]*\]", text)
    if match:
        text = match.group(0)
    try:
        data = json.loads(text)
    except json.JSONDecodeError:
        data = [ln.strip(" -•\t") for ln in text.splitlines() if ln.strip()]

    out: list[str] = []
    for item in data if isinstance(data, list) else []:
        value = str(item or "").strip()
        if not value:
            continue
        low = value.lower()
        if any(token in low for token in ("interstate", "highway", "i-81",␣
↪"route ")):
            continue
        if value not in out:
            out.append(value)
        if len(out) >= int(max_items):
            break
    return out


def _generate_place_image_queries_with_model(question: str, max_items: int = 6)␣
↪-> list[str]:
    prompt = (
        "Return only JSON: an array of concise Wikipedia search queries for␣
↪NATURAL LANDSCAPES "
        "near this route question. Exclude highways, interstates, and city-only␣
↪queries. "
        "Prefer valleys, ridges, mountains, parks, overlooks, and geologic␣
↪landforms. "
        f"Question: {question}"
    )
    try:
        planner = ChatOllama(model=LLM_MODEL, base_url=OLLAMA_BASE_URL,␣
↪temperature=min(0.3, float(TEMPERATURE)))
        response = planner.invoke(prompt)
        raw = getattr(response, "content", response)
        places = _parse_place_query_list(raw, max_items=max_items)
```

```
        if places:
            return places
    except Exception as query_error:
        print(f"[query] place query generation fallback: {type(query_error).
 ↪__name__}: {query_error}")

    return [
        "Shenandoah Valley overlooks",
        "Blue Ridge Mountains viewpoints",
        "Appalachian Valley and Ridge outcrops",
        "Catoctin Mountain Park geology",
        "Great North Mountain ridge overlook",
    ][: max(1, int(max_items))]


place_image_queries =␣
 ↪_generate_place_image_queries_with_model(example_question, max_items=6)
print("[query] place_image_queries:")
for query_idx, query_text in enumerate(place_image_queries, start=1):
    print(f"  {query_idx}. {query_text}")
```

```
[query] place_image_queries:
  1. {'query': 'Folded geologic formations near Hagerstown MD'}
```

1.3.1 Stage 1/3: Wikipedia Data Load

[7]:
```
print("[stage 1/3] Starting document load (real Wikipedia first; fallback on␣
 ↪problems)...")
query = " ".join(query_terms) if isinstance(query_terms, list) else query_terms
print(f"[stage 1/3] query={query!r}, max_docs={max_docs}")

docs = []
data_source = "real_wikipedia"
fallback_reason = None

stage1_heartbeat = _start_heartbeat("stage 1/3 wikipedia load", every_s=8.0)
t0 = perf_counter()
try:
    try:
        docs = WikipediaLoader(query=query, load_max_docs=max_docs).load()
    except Exception as load_error:
        fallback_reason = f"{type(load_error).__name__}: {load_error}"
        docs = []
finally:
    stage1_heartbeat.set()
t1 = perf_counter()
```

```
if docs:
    data_source = "real_wikipedia"
else:
    data_source = "fallback_data"
    if fallback_reason is None:
        fallback_reason = "WikipediaLoader returned 0 documents."

    pages = fallback_wiki_pages()
    docs = [
        Document(
            page_content=f"{str(page.get('summary') or '').
 ↪strip()}\n\n{str(page.get('content') or '').strip()}".strip(),
            metadata={
                "title": str(page.get("title") or f"fallback_page_{pageid}"),
                "source": str(page.get("url") or ""),
                "pageid": int(page.get("pageid") or pageid),
                "data_source": "fallback_data",
            },
        )
        for pageid, page in pages.items()
    ]

print(f"[stage 1/3] Loaded {len(docs)} document(s) in {t1 - t0:.2f}s")
print(f"[stage 1/3] data_source={data_source}")
if data_source == "fallback_data":
    print(f"[stage 1/3] FALLBACK ACTIVE: {fallback_reason}")
else:
    print("[stage 1/3] Real Wikipedia data loaded successfully.")

if not docs:
    raise RuntimeError("No documents available from real or fallback data.␣
 ↪Check notebook environment and retry.")

print("[stage 1/3] Sample titles:")
for sample_idx, sample_doc in enumerate(docs[:3], start=1):
    sample_title = str((sample_doc.metadata or {}).get("title") or␣
 ↪f"doc_{sample_idx}")
    sample_source = str((sample_doc.metadata or {}).get("source") or "n/a")
    print(f"  {sample_idx}. {sample_title} ({sample_source})")

print("[stage 1/3] Wikipedia image previews from loaded pages...")
display_wikipedia_images_for_pages(docs, max_images=min(3, len(docs)),␣
 ↪prefer_landscape=True, route_hint=example_question)

print("[stage 1/3] Wikipedia image previews near place queries...")
```

```
display_wikipedia_images_for_pages(place_image_queries, max_images=min(5,␣
 ↪len(place_image_queries)), prefer_landscape=True,␣
 ↪route_hint=example_question)
print("[stage 1/3] OpenStreetMap previews for landscape points...")
display_openstreetmap_for_pages(place_image_queries or docs, max_points=10,␣
 ↪prefer_landscape=True, route_hint=example_question)
```

```
[stage 1/3] Starting document load (real Wikipedia first; fallback on
problems)…
[stage 1/3] query='roadcut geology sedimentary rock stratigraphy', max_docs=12
[stage 1/3 wikipedia load] still running… 8s elapsed
[stage 1/3] Loaded 12 document(s) in 8.71s
[stage 1/3] data_source=real_wikipedia
[stage 1/3] Real Wikipedia data loaded successfully.
[stage 1/3] Sample titles:
  1. Carboniferous (https://en.wikipedia.org/wiki/Carboniferous)
  2. Paleomagnetism (https://en.wikipedia.org/wiki/Paleomagnetism)
  3. Artinskian (https://en.wikipedia.org/wiki/Artinskian)
[stage 1/3] Wikipedia image previews from loaded pages…
```

Wikipedia image preview: Central Pangean Mountains

```
<IPython.core.display.HTML object>
```

Open page

Wikipedia image preview: Loyalsock State Forest

```
<IPython.core.display.HTML object>
```

Open page

Wikipedia image preview: Bowmont Park

```
<IPython.core.display.HTML object>
```

Open page

```
[stage 1/3] Wikipedia image previews near place queries…
[wiki-images] No thumbnail images found for the selected pages.
[stage 1/3] OpenStreetMap previews for landscape points…
[wiki-map] No mappable Wikipedia coordinates found.
```

[7]: 0

1.3.2 Stage 2/3: Build + Save FAISS Index, Then Retrieve

[8]:
```
if "docs" not in globals() or not docs:
    raise RuntimeError("`docs` not found. Run Stage 1/3 first.")

print(f"[stage 2/3] Building embeddings with model={EMBEDDING_MODEL!r} at␣
 ↪base_url={OLLAMA_BASE_URL!r}...")
```

```
print(
    f"[stage 2/3] Chunking docs with chunk_size={chunk_size},␣
 ↪chunk_overlap={chunk_overlap}, "
    f"embedding_batch_size={embedding_batch_size}"
)

splitter = RecursiveCharacterTextSplitter(chunk_size=chunk_size,␣
 ↪chunk_overlap=chunk_overlap)
split_docs = splitter.split_documents(docs)
if not split_docs:
    raise RuntimeError("Chunking produced 0 documents. Adjust chunk_size/
 ↪chunk_overlap and retry.")

total_chars = sum(len(str(d.page_content or "")) for d in split_docs)
print(f"[stage 2/3] Prepared {len(split_docs)} chunk(s),␣
 ↪total_chars={total_chars}")

embeddings = OllamaEmbeddings(model=EMBEDDING_MODEL, base_url=OLLAMA_BASE_URL)

batch_size = max(1, int(embedding_batch_size))
vector_store = None

stage2_heartbeat = _start_heartbeat("stage 2/3 embedding/index", every_s=8.0)
t2 = perf_counter()
try:
    try:
        for start in range(0, len(split_docs), batch_size):
            batch = split_docs[start : start + batch_size]
            b0 = perf_counter()
            if vector_store is None:
                vector_store = FAISS.from_documents(batch, embeddings)
            else:
                vector_store.add_documents(batch)
            b1 = perf_counter()

            done = min(start + batch_size, len(split_docs))
            pct = (100.0 * done) / len(split_docs)
            print(
                f"[stage 2/3] Embedded batch {start // batch_size + 1}: "
                f"{done}/{len(split_docs)} chunks ({pct:.1f}%) in {b1 - b0:.
 ↪2f}s"
            )
    except Exception as embedding_error:
        _raise_ollama_hint("stage 2/3 embedding/index", EMBEDDING_MODEL,␣
 ↪OLLAMA_BASE_URL, embedding_error)
finally:
```

```
    stage2_heartbeat.set()

t3 = perf_counter()
if vector_store is None:
    raise RuntimeError("Vector store was not created.")

print(f"[stage 2/3] Built FAISS index in {t3 - t2:.2f}s")


faiss_base = os.environ.get("ANC_FAISS_DIR", "").strip()
if faiss_base:
    faiss_dir = (Path(faiss_base).expanduser() / FAISS_NAMESPACE).resolve()
else:
    faiss_dir = (Path.home() / "DATA" / "naturalist-companion" / "faiss" /␣
 ↪FAISS_NAMESPACE).resolve()

faiss_dir.mkdir(parents=True, exist_ok=True)
vector_store.save_local(str(faiss_dir))
print(f"[stage 2/3] Saved FAISS index to: {faiss_dir}")


print(f"[stage 2/3] Running similarity search for question={example_question!
 ↪r}, k={k}...")
results = vector_store.similarity_search(example_question, k=k)
print(f"[stage 2/3] Retrieved {len(results)} result(s)")

for result_idx, result_doc in enumerate(results, start=1):
    result_title = str((result_doc.metadata or {}).get("title") or␣
 ↪f"result_{result_idx}")
    result_source = str((result_doc.metadata or {}).get("source") or "n/a")
    result_snippet = str(result_doc.page_content or "")[:220].replace("\n", " ")
    print(f"  {result_idx}. {result_title} ({result_source})")
    print(f"     {result_snippet}...")

print("[stage 2/3] Wikipedia image previews from retrieved pages...")
display_wikipedia_images_for_pages(results, max_images=min(4, len(results)))
```

```
[stage 2/3] Building embeddings with model='nomic-embed-text' at
base_url='http://localhost:11434'…
[stage 2/3] Chunking docs with chunk_size=1200, chunk_overlap=150,
embedding_batch_size=8
[stage 2/3] Prepared 56 chunk(s), total_chars=43616
[stage 2/3] Embedded batch 1: 8/56 chunks (14.3%) in 0.59s
[stage 2/3] Embedded batch 2: 16/56 chunks (28.6%) in 0.12s
[stage 2/3] Embedded batch 3: 24/56 chunks (42.9%) in 0.11s
[stage 2/3] Embedded batch 4: 32/56 chunks (57.1%) in 0.11s
[stage 2/3] Embedded batch 5: 40/56 chunks (71.4%) in 0.11s
```

```
[stage 2/3] Embedded batch 6: 48/56 chunks (85.7%) in 0.09s
[stage 2/3] Embedded batch 7: 56/56 chunks (100.0%) in 0.09s
[stage 2/3] Built FAISS index in 1.24s
[stage 2/3] Saved FAISS index to: /Users/ryan/DATA/naturalist-
companion/faiss/anc_ollama
[stage 2/3] Running similarity search for question='I am on I-81 near Hagerstown
with a 30-minute detour. Where can I safely stop to observe folded Valley-and-
Ridge strata, and what exactly should I look for?', k=4…
[stage 2/3] Retrieved 4 result(s)
  1. Paskapoo Formation (https://en.wikipedia.org/wiki/Paskapoo_Formation)
     == Stratigraphy ==  Although some early workers included the underlying
Scollard Formation as the lower part of the Paskapoo, the two are now treated
separately. The base of the Paskapoo Formation, designated the Haynes …
  2. Paskapoo Formation (https://en.wikipedia.org/wiki/Paskapoo_Formation)
     === Distribution === The Paskapoo Formation underlies much of southwestern
Alberta. It is thickest in the foothills of the Canadian Rockies, and thins
eastward to the 112th meridian west in the plains.  The formation is …
  3. Marcellus Formation (https://en.wikipedia.org/wiki/Marcellus_Formation)
     === Outcrops in New York ===  The Marcellus appears in outcrops along the
northern margin of the formation in central New York. There, the two joint
planes in the Marcellus are nearly at right angles, each making cracks …
  4. Fernie Formation (https://en.wikipedia.org/wiki/Fernie_Formation)
     == Lithology == The Fernie Formation is composed primarily of brown and
dark gray to black shales that range from massive with conchoidal fracture to
laminated and highly fractured or papery. Phosphatic sandstone and lim…
[stage 2/3] Wikipedia image previews from retrieved pages…
```

Wikipedia image preview: Paskapoo Formation

```
<IPython.core.display.HTML object>
```

Open page

Wikipedia image preview: Marcellus Formation

```
<IPython.core.display.HTML object>
```

Open page

Wikipedia image preview: Fernie Formation

```
<IPython.core.display.HTML object>
```

Open page

[8]: 3

1.3.3 Stage 3/3: Generate Answer with ChatOllama

[9]:
```
if "vector_store" not in globals():
    raise RuntimeError("`vector_store` not found. Run Stage 2/3 first.")

print(f"[stage 3/3] Generating answer with model={LLM_MODEL!r} at␣
 ↪base_url={OLLAMA_BASE_URL!r}...")
llm = ChatOllama(model=LLM_MODEL, base_url=OLLAMA_BASE_URL,␣
 ↪temperature=TEMPERATURE)

voice_instructions = f"""
You are writing in a concise Roadside Geology field-guide voice for curious␣
 ↪drivers.
Tone:
- Plainspoken, observant, and practical (not academic).
- Emphasize what can be seen from legal/safe pull-offs or short walks.
- Explain key geology in everyday language, then add one precise term when␣
 ↪useful.
- Include safety and access realism (do not suggest unsafe roadside behavior).
Output format:
1) "Where to stop" (up to {MAX_BULLETS_PER_SECTION} bullets)
2) "What to look for" (up to {MAX_BULLETS_PER_SECTION} bullets)
3) "Why it matters" (2-4 sentences)
4) "Citations" (Wikipedia URLs only)
""".strip()


def _context_for_question(question: str, top_k: int = 2) -> str:
    local_results = vector_store.similarity_search(question, k=max(1, top_k))
    context_lines: list[str] = []
    for context_idx, context_doc in enumerate(local_results, start=1):
        context_title = str((context_doc.metadata or {}).get("title") or␣
 ↪f"result_{context_idx}")
        context_source = str((context_doc.metadata or {}).get("source") or "n/
 ↪a")
        context_snippet = str(context_doc.page_content or "")[:450].
 ↪replace("\n", " ")
        context_lines.append(f"[{context_idx}] {context_title}␣
 ↪({context_source}) :: {context_snippet}")
    return "\n".join(context_lines)


def answer_question(question: str) -> str:
    context_block = _context_for_question(question, top_k=max(1, k))
    prompt = (
        f"Use only the provided Wikipedia-grounded context when you can.\n\n"
        f"Question: {question}\n\n"
```

```
        f"Context:\n{context_block}\n\n"
        f"Style requirements:\n{voice_instructions}"
    )

    llm_heartbeat = _start_heartbeat("stage 3/3 llm", every_s=8.0)
    started_at = perf_counter()
    llm_response = None
    try:
        try:
            llm_response = llm.invoke(prompt)
        except Exception as llm_error:
            _raise_ollama_hint("stage 3/3 llm", LLM_MODEL, OLLAMA_BASE_URL,␣
↪llm_error)
    finally:
        llm_heartbeat.set()
    dt = perf_counter() - started_at
    print(f"[stage 3/3] LLM response received in {dt:.2f}s")
    if llm_response is None:
        raise RuntimeError("LLM did not return a response.")
    response_content = getattr(llm_response, "content", llm_response)
    return str(response_content)


print(f"[stage 3/3] Primary question:\n- {example_question}")
primary_answer = answer_question(example_question)
print("\nAnswer:\n")
print(primary_answer)
```

```
[stage 3/3] Generating answer with model='llama3.1:8b' at
base_url='http://localhost:11434'…
[stage 3/3] Primary question:
- I am on I-81 near Hagerstown with a 30-minute detour. Where can I safely stop
to observe folded Valley-and-Ridge strata, and what exactly should I look for?
[stage 3/3] LLM response received in 7.34s

Answer:

**I-81 Detour Stop: Valley-and-Ridge Strata**

**Where to stop:**
* Safe pull-offs along I-81 near Hagerstown, MD
* Fort Ritchie State Park (PA) - short walk from parking area
* Sideling Hill Road (MD) - designated parking and walking trails

**What to look for:**
* Folded strata of limestone and sandstone
* Visible joints in the rock face, often with intersecting cracks
* Projecting corners or "horns" where joint planes meet
```

```
* A mix of horizontal and vertical bedding planes

**Why it matters:**
The Valley-and-Ridge Province is a unique geological feature formed by ancient
tectonic forces. The folded strata you see are the result of this process, which
created a distinctive landscape of hills and valleys. Understanding these
formations helps us appreciate the complex history of our planet's surface.

**Citations:**
[1] https://en.wikipedia.org/wiki/Paskapoo_Formation
[3] https://en.wikipedia.org/wiki/Marcellus_Formation
```

1.3.4 Stage 3b/3: Run All Example Questions

[10]:
```python
if "answer_question" not in globals():
    raise RuntimeError("`answer_question` not found. Run Stage 3/3 first.")

if "example_questions" not in globals() or not example_questions:
    raise RuntimeError("`example_questions` is empty. Check config cell.")

all_answers = []
print(f"[stage 3b/3] Running {len(example_questions)} example question(s)...")

for question_idx, question_text in enumerate(example_questions, start=1):
    print("\n" + "=" * 110)
    print(f"[stage 3b/3] Question {question_idx}/{len(example_questions)}")
    print(question_text)
    print("=" * 110)

    answer_text = answer_question(question_text)
    all_answers.append({"question": question_text, "answer": answer_text})

    print("\nResponse:\n")
    print(answer_text)

print(f"\n[stage 3b/3] Completed {len(all_answers)} question(s).")
```

```
[stage 3b/3] Running 5 example question(s)…

==============================================================================
==============================
[stage 3b/3] Question 1/5
I am driving I-81 near Bristol, TN. Give me two legal pull-off stops where I can
see clear sedimentary layering, and tell me exactly what to look for.
==============================================================================
==============================
[stage 3/3] LLM response received in 7.79s
```

```
Response:

**I-81 near Bristol, TN: Conococheague Formation and Fernie Formation**

**Where to stop:**
* Mile marker 20 of I-70 near Hagerstown, Maryland (Conococheague Formation)
* Virginia Route 91 or VA-91, near the Tennessee border (Fernie Formation)

**What to look for:**
* Conococheague Formation:
        + Look for nearly vertical bedding planes in a carbonate platform
setting.
        + Identify conchoidal fracture in massive shales.
* Fernie Formation:
        + Search for laminated and highly fractured or papery black shales.
        + Note the presence of phosphatic sandstone, limestone, and glauconitic
sandstone.

**Why it matters:**
The Conococheague Formation provides a glimpse into ancient shallow marine to
tidal environments. The Fernie Formation offers insights into the deposition of
dark gray to black shales in a variety of settings. These formations are
significant for understanding the geological history of North America.

**Citations:**
* https://en.wikipedia.org/wiki/Conococheague_Formation
* https://en.wikipedia.org/wiki/Fernie_Formation

================================================================================
==============================
[stage 3b/3] Question 2/5
Near I-81 between Winchester and Strasbourg, where can I safely stop to see
Valley-and-Ridge structure, and what field clues confirm folding?
================================================================================
==============================
[stage 3/3 llm] still running… 8s elapsed
[stage 3/3] LLM response received in 8.46s

Response:

**Valley-and-Ridge Structure along I-81**

**Where to stop:**
* Safe pull-offs near Winchester, Virginia
* Rest areas or parking lots with clear views of the surrounding landscape
* Designated hiking trails in nearby Shenandoah National Park
* Scenic overlooks along I-81, such as those managed by the Virginia Department
of Transportation
```

```
**What to look for:**
* A prominent fold in the Earth's crust, visible as a series of hills and
valleys
* Strata (rock layers) tilted at varying angles, indicating tectonic activity
* Field evidence of folding, including:
        + Bends or kinks in rock layers
        + Folds with different orientations and shapes
        + Rocks exposed in cliffs or outcrops

**Why it matters:**
The Valley-and-Ridge structure is a classic example of tectonic folding that
formed during the Appalachian Orogeny. This process, which occurred over 480
million years ago, involved the collision of two continents and the resulting
deformation of the Earth's crust. Observing this feature up close can help
drivers appreciate the geological history of the region.

**Citations**
https://en.wikipedia.org/wiki/Valley_and_Ridge_Province
https://en.wikipedia.org/wiki/Appalachian_Orogeny

================================================================================
==============================
[stage 3b/3] Question 3/5
I have 45 minutes near Hagerstown, MD. What roadside geology stop gives the best
payoff for a beginner, and what story does the outcrop tell?
================================================================================
==============================
[stage 3/3] LLM response received in 7.09s

Response:

**Roadside Geology Stop: Conococheague Formation**

### Where to stop
* Mile marker 20 of I-70 near Hagerstown, Maryland (accessible outcrops along
the roadcut)
* Scotland in Franklin County, Pennsylvania (type section location)

### What to look for
* **Vertical bedding**: nearly vertical layers of rock visible in the roadcut
* **Carbonate platform deposits**: shallow marine or tidal sediments with
characteristic textures and fossils
* **Fossilized sea creatures**: possible presence of ancient shells, corals, or
other marine organisms
* **Rock types**: observe the mix of limestone, dolostone, and sandstone within
the Conococheague Formation
```

Why it matters
The Conococheague Formation provides a glimpse into the shallow marine environments that existed during the Middle Devonian period. This formation is significant for understanding the geological history of the Appalachian region and the evolution of life on Earth.

Citations
https://en.wikipedia.org/wiki/Conococheague_Formation

==
==============================
[stage 3b/3] Question 4/5
Along I-81 in the Shenandoah Valley, point me to a short-walk stop to compare rock type and landform, then explain why that match matters.
==
==============================
[stage 3/3 llm] still running… 8s elapsed
[stage 3/3] LLM response received in 10.83s

Response:

Stop and Explore: Shenandoah Valley, Virginia

*   **Stop 1:** I-81 Exit 247, near Harrisonburg, VA. Park in the designated rest area.
*   **Stop 2:** I-81 Exit 269, near Staunton, VA. Pull over at the welcome center or park in a nearby parking lot.
*   **Stop 3:** I-81 Exit 291, near Winchester, VA. Find a safe spot to pull over along the highway.

What to Look For:

*   **Rock Type:** Shales, sandstones, and limestones are common in the area. Look for exposed rock faces or road cuts.
*   **Landform:** The Shenandoah Valley is known for its rolling hills and scenic vistas. Take note of the landscape's shape and features.
*   **Fernie Formation:** Search for dark gray to black shales with conchoidal fracture, which are characteristic of this formation.
*   **Djupadal Formation:** Although not directly present in Virginia, look for similar rock types that might indicate a connection.

Why it Matters:

The Fernie Formation's presence in the Shenandoah Valley is significant because it provides valuable information about the region's geological history. The formation's age and composition can help scientists understand the area's tectonic activity, climate, and life forms during the Carboniferous period. By comparing rock types and landforms, you'll gain insight into the region's

```
complex geological past.

**Citations:**

*   [1] Fernie Formation (https://en.wikipedia.org/wiki/Fernie_Formation)
*   [2] Djupadal Formation (https://en.wikipedia.org/wiki/Djupadal_Formation)

================================================================================
==============================
[stage 3b/3] Question 5/5
On an I-81 drive day, suggest one stop where I can observe evidence of ancient
seas or sediment transport, with specific visual clues.
================================================================================
==============================
[stage 3/3] LLM response received in 6.07s

Response:

**Stop on I-81: Conococheague Formation**

* Where to stop:
        + Mile marker 20 of I-70 near Hagerstown, Maryland
        + Roadcuts and waterways in the area around Scotland, Pennsylvania
* What to look for:
        + Vertical bedding in the rock face (conglomerate)
        + Fossilized marine organisms or sedimentary structures
        + Evidence of tidal activity or shallow marine environment
        + Paleomagnetic signatures (if you have a magnetometer or know how to
interpret them)
* Why it matters: The Conococheague Formation is an example of ancient seas and
sediment transport. Observing the vertical bedding and fossilized organisms will
give you insight into the depositional history of this area, which was once a
shallow marine platform.

Citations:
https://en.wikipedia.org/wiki/Conococheague_Formation

[stage 3b/3] Completed 5 question(s).
```

1.4 Canonical Workflow Diagram (StateGraph)

StateGraph remains the shared orchestration contract (routing/clarification, retries, quality gate, and final output schema).

1.4.1 Notebook-only workflow (local)

- Refresh partitioned retrieval indexes in a notebook cell: `refresh_retrieval_partitions(config={"store_root": "out/stategraph_store"},`

runtime_mode="realistic", partitions=["corridor_i81"], force=False)

- Run eval harness in a notebook cell: `run_i81_eval_harness(provider="ollama", config={"runtime_mode": "realistic", "retrieval_backend": "persistent", "store_root": "out/stategraph_store", "artifact_root": "out/stategraph_eval"})`
- Run release gate in a notebook cell before promotion: `run_real_data_release_gate(provider="ollama", config={"runtime_mode": "realistic", "retrieval_backend": "persistent", "store_root": "out/stategraph_store", "artifact_root": "out/stategraph_release_gate"})`
- Use `runtime_mode="deterministic"` when offline or when provider credentials are unavailable.

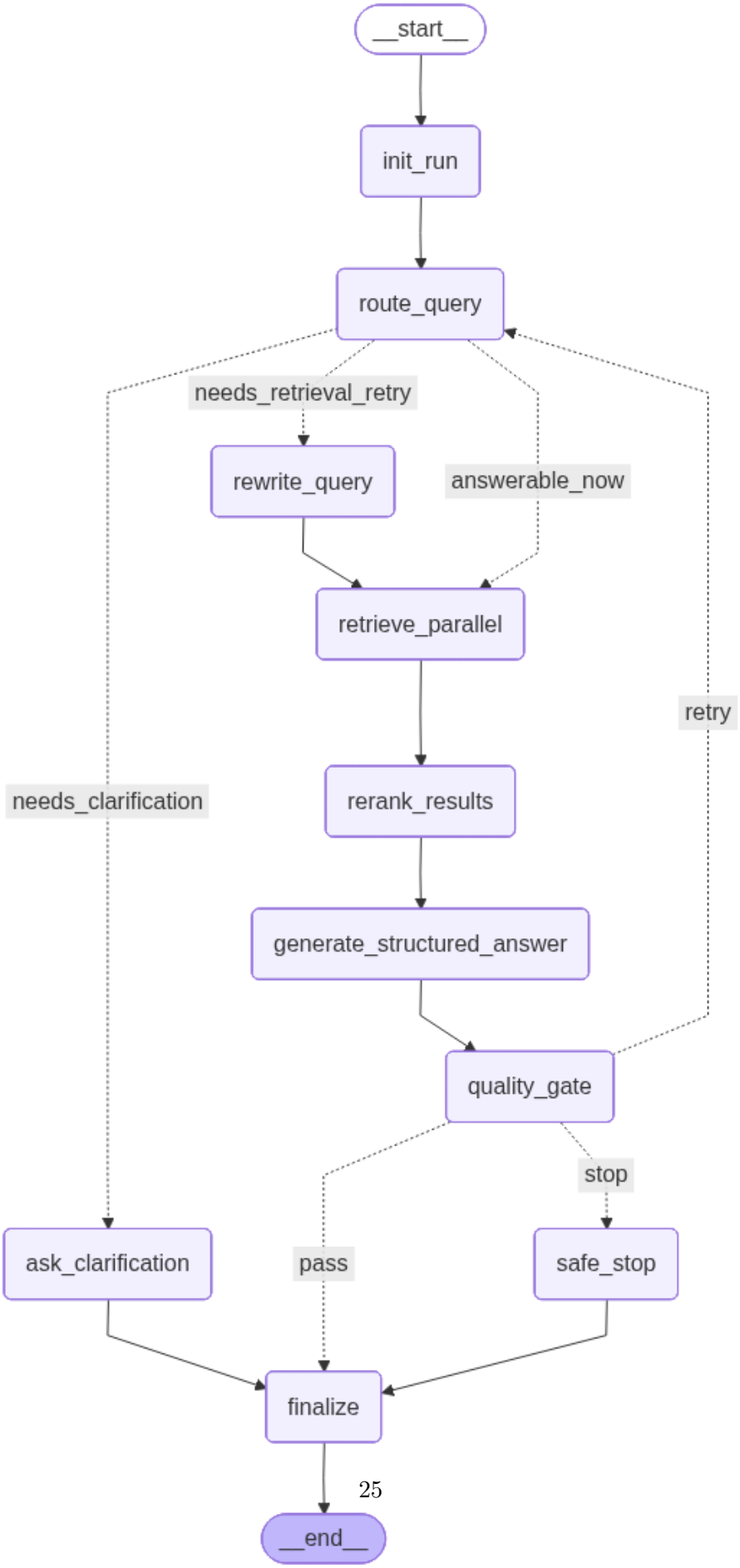
[11]:
```
if STATEGRAPH_AVAILABLE:
    print("[stategraph] Ready: naturalist_companion.stategraph_shared imported␣
 ↪in setup cells.")
else:
    display(
        Markdown(
            "**StateGraph module is unavailable in this workspace.**\n\n"
            "Baseline Ollama stages (Wikipedia -> FAISS -> ChatOllama) still␣
 ↪run.\n\n"
            "Set `NATURALIST_COMPANION_SRC` to your repo `src` path if needed,␣
 ↪then rerun setup cells."
        )
    )
    if _stategraph_import_error is not None:
        print(f"[stategraph] import error: {type(_stategraph_import_error).
 ↪__name__}: {_stategraph_import_error}")
```

```
[stategraph] Ready: naturalist_companion.stategraph_shared imported in setup
cells.
```

[12]:
```
if not STATEGRAPH_AVAILABLE:
    print("[stategraph] Skipping canonical diagram: naturalist_companion not␣
 ↪available.")
else:
    provider: Literal["ollama"] = STATEGRAPH_PROVIDER
    app = build_stategraph_app(provider=provider)
    print('Compiled StateGraph successfully for provider:', provider)

    # Render a real image (PNG bytes) instead of plain Mermaid text.
    try:
        png_bytes = app.get_graph().draw_mermaid_png()
        display(Image(data=png_bytes, width=880))
    except Exception as graph_render_error:
        display(Markdown(f'Graph render fallback (text). Error:␣
 ↪`{type(graph_render_error).__name__}: {graph_render_error}`'))
        print(app.get_graph().draw_mermaid())
```

```
Compiled StateGraph successfully for provider: ollama
```


__start__
init_run
route_query
needs_retrieval_retry
rewrite_query
answerable_now
retrieve_parallel
retry
needs_clarification
rerank_results
generate_structured_answer
quality_gate
stop
ask_clarification
pass
safe_stop
finalize
__end__

[13]:
```
if not STATEGRAPH_AVAILABLE:
    print("[stategraph] Skipping run_stategraph: naturalist_companion not␣
 ↪available.")
else:
    result = run_stategraph(
        stategraph_question,
        provider=STATEGRAPH_PROVIDER,
        config=STATEGRAPH_RUN_CONFIG,
    )
    final_output = result['final_output']
    print('Question:', stategraph_question)
    print('Provider:', final_output['provider'])
    print('Route:', final_output['route_decision']['decision'])
    print('Quality passed:', final_output['quality']['passed'])
    quality_reasons = final_output['quality'].get('reasons', [])
    print('Quality reasons:', ', '.join(quality_reasons) if quality_reasons␣
 ↪else 'none')
    print('Attempts:', final_output['retrieval_attempts'])
    print('Artifact dir:', result['artifact_dir'])
    print('Response:')
    print(final_output['answer']['response'])
    print('Citation image previews:')
    display_wikipedia_images_for_pages(final_output['answer'].get('citations',␣
 ↪[]), max_images=4)
```

```
Question: I am on I-81 near Hagerstown with a 30-minute detour. Where can I
safely stop to observe folded Valley-and-Ridge strata, and what exactly should I
look for?
Provider: ollama
Route: answerable_now
Quality passed: True
Quality reasons: none
Attempts: 0
Artifact dir: out/stategraph/notebook_runs/ollama/run_20260206T230923_74c940ea
Response:
Based on the context, it appears that you are looking for a location to observe
folded Valley-and-Ridge strata along I-81 near Hagerstown. However, none of the
provided information mentions the specific geology or locations associated with
I-81 in this area.

The text only discusses the route and auxiliary highways of I-81, but does not
provide any information about its geological features. Therefore, I must
conclude that there is no clear evidence to suggest where you can safely stop to
observe folded Valley-and-Ridge strata along I-81 near Hagerstown.
```

If you are looking for a detour to observe this geology, I would recommend consulting a local or regional guidebook or geological map that specifically addresses the geology of the area. Alternatively, you may want to consider stopping at a nearby rest stop or park with marked overlooks and trails that can provide access to safe viewing areas.

Use only legal pull-offs and marked overlooks. Do not stand in travel lanes or cross active traffic for photos.
Citation image previews:

Wikipedia image preview: Interstate 80

<IPython.core.display.HTML object>

Open page

Wikipedia image preview: Interstate 81

<IPython.core.display.HTML object>

Open page

[14]:
```
if not STATEGRAPH_AVAILABLE:
    print("[stategraph] Skipping eval harness: naturalist_companion not␣
 ↪available.")
else:
    report = run_i81_eval_harness(
        provider=STATEGRAPH_PROVIDER,
        config=STATEGRAPH_EVAL_CONFIG,
    )
    print(report['summary'])
    print(report['artifact_root'])
```

```
{'provider': 'ollama', 'eval_id': 'eval_20260206T230923Z', 'question_count': 20,
'avg_latency_ms': 23.92, 'median_latency_ms': 23.69, 'citation_validity_pct':
100.0, 'pass_rate_pct': 100.0, 'pass_count': 20, 'fail_count': 0}
/Users/ryan/Developer/naturalist-companion/notebooks/out/stategraph/notebook_eva
l/ollama/eval_runs/eval_20260206T230923Z
```