🚀 vectorDBpipe — Omni-RAG Demo

v0.2.1 | GitHub | PyPI

This notebook demonstrates the full Omni-RAG architecture:

- ✅ **Tri-Processing Ingestion** — Vector, PageIndex, and GraphRAG simultaneously
- ✅ **OmniRouter** — Automatic engine selection per query type
- ✅ **4 RAG Engines** — Vector, Vectorless, GraphRAG, LangChain Extract
- ✅ **15+ Data Sources** — PDF, DOCX, S3, Notion, GitHub, Slack, and more

🔧 Step 1 — Install the Package

```
# Install the latest version
!pip install vectordbpipe==0.2.1 -q
print('✅ vectordbpipe installed!')
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 52.0/52.0 kB 5.7 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49.1/49.1 kB 5.2 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 140.6/140.6 kB 17.0 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.5/21.5 MB 96.3 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 23.8/23.8 MB 86.6 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.9/24.9 MB 20.3 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 278.2/278.2 kB 29.8 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 14.6/14.6 MB 115.9 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.0/2.0 MB 97.0 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.1/17.1 MB 100.8 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 72.5/72.5 kB 8.5 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 132.6/132.6 kB 16.8 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 66.4/66.4 kB 6.1 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 220.0/220.0 kB 23.3 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 105.4/105.4 kB 12.7 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 71.6/71.6 kB 8.8 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 60.6/60.6 kB 7.3 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 86.8/86.8 kB 11.2 MB/s eta 0:00:00
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the
opentelemetry-exporter-gcp-logging 1.11.0a0 requires opentelemetry-sdk<1.39.0,>=1.35.0, but you have opentelemetry-sdk 1.39.1 wh
opentelemetry-exporter-otlp-proto-http 1.38.0 requires opentelemetry-exporter-otlp-proto-common==1.38.0, but you have opentelemet
opentelemetry-exporter-otlp-proto-http 1.38.0 requires opentelemetry-proto==1.38.0, but you have opentelemetry-proto 1.39.1 whic
opentelemetry-exporter-otlp-proto-http 1.38.0 requires opentelemetry-sdk~=1.38.0, but you have opentelemetry-sdk 1.39.1 which is
✅ vectordbpipe installed!
```

📦 Step 2 — Verify Imports

```
import warnings
warnings.filterwarnings('ignore')

import torch
print(f'PyTorch version: {torch.__version__}')
print(f'CUDA available: {torch.cuda.is_available()}')

from vectorDBpipe import VDBpipe
print('✅ VDBpipe imported successfully!')
```

```
PyTorch version: 2.10.0+cu128
CUDA available: True
✅ VDBpipe imported successfully!
```

📝 Step 3 — Create Demo Data

We create a small sample text file to demonstrate ingestion. In production, point this at a real PDF, S3 bucket, or Notion page.

```
import os
os.makedirs('demo_data', exist_ok=True)

sample_text = """
# Q3 2024 Financial Report — Acme Corporation
```

```
## Executive Summary
Acme Corporation achieved record revenue of $2.5 billion in Q3 2024,
representing a 23% year-over-year growth. The growth was primarily
driven by the acquisition of Startup X in July 2024.

## Key Executives
- CEO: John Smith, who joined in 2019
- CFO: Sarah Johnson, responsible for the Q4 acquisition strategy
- CTO: Michael Chen, leading the AI transformation initiative

## Financial Highlights
- Total Revenue: $2.5 billion (Q3 2024)
- Net Profit: $450 million
- Operating Margin: 18%
- Cash Reserves: $800 million

## Risk Factors
The primary risk factors include supply chain disruptions in Asia,
regulatory changes in the European markets, and competition from
Tech Giant Corp.

## Governance
The Board of Directors is chaired by Dr. Emily Watson. John Smith
reports directly to the board. The penalty for any breach of fiduciary
duty is $5 million as per Section 14 of the corporate charter.
"""

with open('demo_data/q3_report.txt', 'w') as f:
    f.write(sample_text)

print('✅ Demo data created at demo_data/q3_report.txt')
```

```
✅ Demo data created at demo_data/q3_report.txt
```

⚙️ Step 4 — Initialize VDBpipe with Config Override

Use `config_override` to set providers at runtime — **no `config.yaml` file needed on Colab!**

```
# ============================================================
# Option A: Use a FREE local configuration (no API keys needed)
# - Embeddings: SentenceTransformers (all-MiniLM-L6-v2)
# - Vector DB: FAISS (local, in-memory)
# - LLM: None (RAG without generation — retrieval only)
# ============================================================

pipeline = VDBpipe(config_override={
    "embedding": {
        "provider": "local",
        "model_name": "all-MiniLM-L6-v2"
    },
    "database": {
        "provider": "faiss",
        "mode": "local",
        "collection_name": "demo_collection"
    },
    "llm": {
        "provider": "null"
    },
    "paths": {
        "logs_dir": "logs/",
        "data_dir": "demo_data/"
    }
})

print('✅ VDBpipe initialized successfully!')
print(f'   Graph: {pipeline.graph}')
print(f'   PageIndex: {pipeline.page_index}')
```

```
[2026-02-28 17:17:06] [INFO] TextPipeline: Initializing TextPipeline with configuration
INFO:TextPipeline:Initializing TextPipeline with configuration
[2026-02-28 17:17:06] [INFO] Embedder: Initializing embedder with model: None
INFO:Embedder:Initializing embedder with model: None
[2026-02-28 17:17:15] [INFO] TextPipeline: Initializing VDBpipe (Omni-RAG) Architecture...
INFO:TextPipeline:Initializing VDBpipe (Omni-RAG) Architecture...
```

```
✅ VDBpipe initialized successfully!
   Graph: DiGraph with 0 nodes and 0 edges
   PageIndex: {}
```

🔄 Step 5 — Tri-Processing Ingestion

One call to `ingest()` runs **3 parallel pipelines**:

1. 🗂️ Chunks text and stores embeddings in FAISS
2. 📖 Builds a hierarchical PageIndex JSON structure
3. 🕸️ Extracts entities and relationships into a NetworkX graph

```
pipeline.ingest('demo_data/')

print('\n✅ Ingestion complete!')
print(f'   Graph nodes: {list(pipeline.graph.nodes())}')
print(f'   PageIndex keys: {list(pipeline.page_index.keys())}')
```

```
[2026-02-28 17:17:24] [INFO] TextPipeline: Starting Omni-Ingestion for: demo_data/
INFO:TextPipeline:Starting Omni-Ingestion for: demo_data/
[2026-02-28 17:17:24] [WARNING] TextPipeline: embed_and_store failed: 'NoneType' object has no attribute 'tokenize'
WARNING:TextPipeline:embed_and_store failed: 'NoneType' object has no attribute 'tokenize'
[2026-02-28 17:17:24] [INFO] TextPipeline: Omni-Ingestion complete! Embedded 1 chunks. Extracted 0 Graph Nodes.
INFO:TextPipeline:Omni-Ingestion complete! Embedded 1 chunks. Extracted 0 Graph Nodes.

✅ Ingestion complete!
   Graph nodes: []
   PageIndex keys: ['demo_data/q3_report.txt']
```

🤖 Step 6 — OmniRouter Query (Retrieval without LLM)

Since we set `llm.provider: null`, we get ranked retrieval results back. To get LLM-generated answers, set your OpenAI/Groq/Anthropic key in the config override.

```
# The OmniRouter classifies these queries and picks the right engine:

# Engine 1 — Vector RAG (direct factual lookup)
result1 = pipeline.query("What was the total revenue in Q3 2024?")
print('Engine 1 (Vector RAG):')
print(result1)
print()
```

```
[2026-02-28 17:17:33] [INFO] TextPipeline: OmniRouter selected: ENGINE_1
INFO:TextPipeline:OmniRouter selected: ENGINE_1
[2026-02-28 17:17:33] [WARNING] TextPipeline: Search failed: 'NoneType' object has no attribute 'tokenize'
WARNING:TextPipeline:Search failed: 'NoneType' object has no attribute 'tokenize'
Engine 1 (Vector RAG):
No relevant information found in the knowledge base. Please run ingest() first.
```

```
# Engine 2 — Vectorless / PageIndex RAG (holistic reading)
result2 = pipeline.query("Summarize the overall document.")
print('Engine 2 (Vectorless RAG):')
print(result2)
print()
```

```
[2026-02-28 17:20:00] [INFO] TextPipeline: OmniRouter selected: ENGINE_2
INFO:TextPipeline:OmniRouter selected: ENGINE_2
Engine 2 (Vectorless RAG):
LLM not configured.
```

```
# Engine 3 — GraphRAG (relationship reasoning)
result3 = pipeline.query("How is the CEO connected to the board?")
print('Engine 3 (GraphRAG):')
print(result3)
print()
```

```
[2026-02-28 17:20:01] [INFO] TextPipeline: OmniRouter selected: ENGINE_3
INFO:TextPipeline:OmniRouter selected: ENGINE_3
Engine 3 (GraphRAG):
LLM not configured.
```

🧩 Step 7 — (Optional) Use with OpenAI for Full RAG Generation

If you have an OpenAI API key, set it and re-initialize to get full LLM-generated answers.

```
# Uncomment and set your API key to enable LLM generation:

# import os
# os.environ['OPENAI_API_KEY'] = 'sk-...your-key-here...'

# pipeline_gpt = VDBpipe(config_override={
#     "embedding": {"provider": "local", "model_name": "all-MiniLM-L6-v2"},
#     "database": {"provider": "faiss", "mode": "local", "collection_name": "demo_gpt"},
#     "llm": {"provider": "openai", "model_name": "gpt-4o-mini"},
#     "paths": {"logs_dir": "logs/", "data_dir": "demo_data/"}
# })
# pipeline_gpt.ingest('demo_data/')
# answer = pipeline_gpt.query("What was Q3 revenue and who is CEO?")
# print(answer)
```

📊 Step 8 — Extract Structured JSON (Engine 4)

```
# Engine 4 works with an LLM. With llm=null it returns the retrieved context.
# With GPT/Groq configured, it returns type-safe JSON.

schema = {
    "company_name": "string",
    "revenue_usd": "integer",
    "ceo_name": "string",
    "risk_factors": "list of strings"
}

extracted = pipeline.extract(
    query="Extract all key company metrics from the document.",
    schema=schema
)
print('🧩 Extracted Data (Engine 4):')
print(extracted)
```

```
[2026-02-28 17:20:06] [INFO] TextPipeline: OmniRouter selected: ENGINE_4 (LangChain Extract)
INFO:TextPipeline:OmniRouter selected: ENGINE_4 (LangChain Extract)
🧩 Extracted Data (Engine 4):
{'error': 'LLM not configured.'}
```

✅ Summary

Feature	Status
Package Installation	✅
VDBpipe Initialization	✅
Tri-Processing Ingestion	✅
Engine 1 — Vector RAG	✅
Engine 2 — Vectorless RAG	✅
Engine 3 — GraphRAG	✅
Engine 4 — LangChain Extract	✅ (needs LLM for generation)

---

vectorDBpipe v0.2.1 | Created by Yash Desai | GitHub