VOLTRON DATA

Theseus for RAG Workflows

Maximize Data Freshness and Tokenize on the Fly

Embed vector search right in SQL, enabling familiar query patterns to unleash GPU-accelerated petabyte-scale retrieval for blazing-fast, hassle-free RAG performance.

Maximize Data Freshness and Tokenize on the Fly	Adopt scalable AI pipelines using SQL
Running your RAG pipeline at enterprise scale, data (code, documents, presentations, etc.) is constantly changing, and has to be constantly retokenized. This takes massive GPU datacenters hours to days to compute. And like driving a new car off the lot, it's immediately lost 20% of its value. What if instead of tokenizing all of your data ahead, you just did it runtime? What if you can use the freshest data without the overhead?	Embed vector search pipelines directly into standard SQL workflows to seamlessly integrate powerful LLM capabilities within critical data analytics pipelines. Integrating SQL-based Retrieval-Augmented Generation (RAG) helps data engineers deliver more precise, timely business insights at significantly lower cost and complexity. Embedding vector search and retrieval capabilities directly in SQL eliminates costly, inefficient external data pipelines and complex orchestration, improving performance and reducing operational overhead and compute costs.

On-demand Vectorization In-Situ with Theseus

Unlike traditional approaches that pre-compute embeddings for entire datasets, Theseus performs on-demand vectorization, generating embeddings only when needed. This approach significantly reduces computational overhead while maintaining rapid response times.

- **Seamless Vector Search in SQL**
  Embed vector similarity search as a **user-defined function (UDF)** inside SQL queries with no external pipelines or orchestration required.
- **Run at Petabyte Scale**
  Thanks to GPU acceleration, Theseus can process structured and semi-structured datasets **measured in petabytes**, delivering results in seconds, not hours.
- **Feed LLMs with Structured Context**
  Use real-time production data as context for LLM responses. Your models now generate **domain-specific, up-to-date insights** using SQL queries on live data.
- **Reduce Infrastructure and Ops Overhead**
  No need to stitch together Python libraries, vector DBs, and APIs. Fewer moving parts mean faster performance and lower compute costs.
- **Efficient data location**
  Theseus orchestrates multi-step querying across knowledge graphs, structured data, and vector spaces for LLMs to receive precisely curated information to generate high-quality responses.

Drawbacks of Traditional RAG Approaches

Traditional RAG pipelines rely heavily on Python orchestration, vector databases (e.g., Pinecone, FAISS, Cassandra, Chroma), making it complex to integrate into production SQL environments, leading to inefficiencies at scale, and increasing retrieval costs.

Traditional RAG pipeline performance significantly deteriorates when introducing complex operations such as joins, sorts, aggregations, or filters across multiple retrieval sources.

The Advantages of Theseus

Theseus leverages SQL-native vector search and GPU-accelerated query performance for production-scale, structured, and performance-critical applications.

	Theseus	Others
Retrieval Method	SQL dialect, structured & vector	Similarity search with embeddings
Infrastructure	GPU-accelerated, SQL-native engine	Python libraries, vector databases
Scale	Petabyte scale, structured and semi-structured	Document-level, small-to-medium scale
Target User	Data engineers, SQL analysts	AI developers, data scientists
Use Cases	Enterprise analytics, SQL pipelines	Document retrieval, chatbots, QA systems

Build RAG Pipelines with SQL Statements

Python

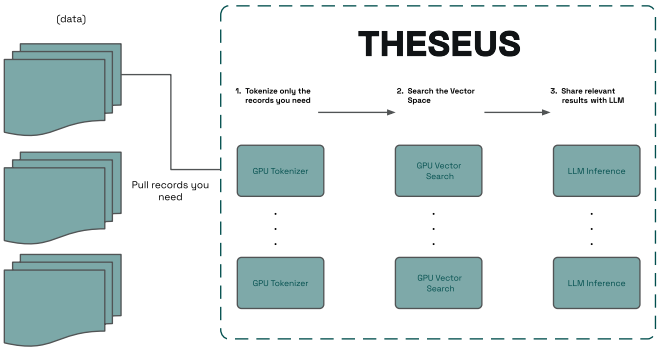
```python
k = 100
user_question = ""“Where are earthquakes causing damage?”""

result = con.sql(f"""
SELECT
    source_url, source_text,  rag.find_neares-t_neighbor_distances(
        embedding, '{user_question}'
    ) AS distance_result
FROM gdelt_text_embeddings
ORDER BY
distance_result ASC
LIMIT {k}
""").to_pyarrow()
agent_response = chat.ask(user_question, result['source_text'])

pprint(agent_response[0].as_py())
```

Native SQL Integration
Utilize familiar SQL query patterns to seamlessly incorporate RAG techniques.

Optimized for Scale
Efficiently process petabyte-scale datasets with integrated GPU acceleration.

Performance Efficiency
Built-in query optimization reduces complexity and ensures high-performance retrieval without manual tuning.

Example RAG Pipeline with JIT Tokenization and Embedding



- **Pull in RAW data into GPU memory (CSV, Parquet, JSON)**
  for example, news articles with metadata and URL to news source.
- **Generate embeddings in situ**
  scrape text from news articles and generate embeddings with a GPU tokenizer using tools like Hugging Face and Triton Inference
- **Search embeddings in situ**
  use a vector search tool or library like Pinecone, Quadrant, AstraDB or NVIDIA cuVS to search embeddings for articles relevant to the question asked.
- **Inference/LLM in situ**
  feed relevant articles alongside the user question and generate a response using Langchain, RaySegve or AWS Bedrock.

Try Theseus Today.

Learn more: https://voltrondata.com/theseus ↗

Contact us : https://voltrondata.com/contact ↗