ExtremeXP Knowledge Graph System

Automating Scientific Literature Analysis

Erik Pahor

Project Presentation

June 13, 2025

Outline

1.1 The Problem: A Data Deluge

- Massive volume of scientific papers.

1.1 The Problem: A Data Deluge

- Massive volume of scientific papers.
- Metadata is unstructured and siloed.

1.1 The Problem: A Data Deluge

- Massive volume of scientific papers.
- Metadata is unstructured and siloed.
- Discovering trends and relations is difficult and manual.

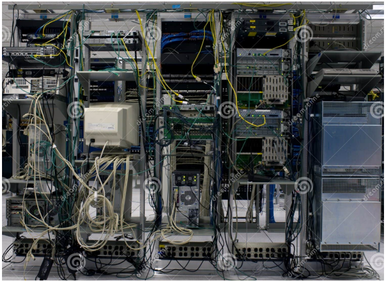


Figure: Messy, disconnected data points.

Problem Statement

How can we transform disparate paper metadata into a structured, queryable knowledge base to accelerate scientific discovery?

1.2 Project Description: The Solution

Our Goal

To build a **production-ready, automated pipeline** to construct and manage a knowledge graph of scientific papers.

1.2 Project Description: The Solution

Our Goal

To build a **production-ready, automated pipeline** to construct and manage a knowledge graph of scientific papers.

Key Features:

- **Automated Ingestion:** Processes JSON via API and file monitoring.

1.2 Project Description: The Solution

Our Goal

To build a **production-ready, automated pipeline** to construct and manage a knowledge graph of scientific papers.

Key Features:

- **Automated Ingestion:** Processes JSON via API and file monitoring.
- **Intelligent Processing:** Validates, normalizes, and enriches data.

1.2 Project Description: The Solution

Our Goal

To build a **production-ready, automated pipeline** to construct and manage a knowledge graph of scientific papers.

Key Features:

- **Automated Ingestion:** Processes JSON via API and file monitoring.
- **Intelligent Processing:** Validates, normalizes, and enriches data.
- **Robust Storage:** Uses an industry-standard RDF triplestore.

1.2 Project Description: The Solution

Our Goal

To build a **production-ready, automated pipeline** to construct and manage a knowledge graph of scientific papers.

Key Features:

- **Automated Ingestion:** Processes JSON via API and file monitoring.
- **Intelligent Processing:** Validates, normalizes, and enriches data.
- **Robust Storage:** Uses an industry-standard RDF triplestore.
- **Full Observability:** Built-in health checks, metrics, and logging.

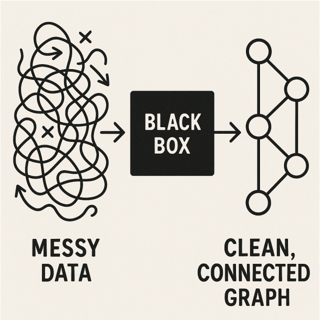


Figure: Messy data entering our system and emerging as a clean, connected graph.

1.3 System Architecture

A Microservices Approach

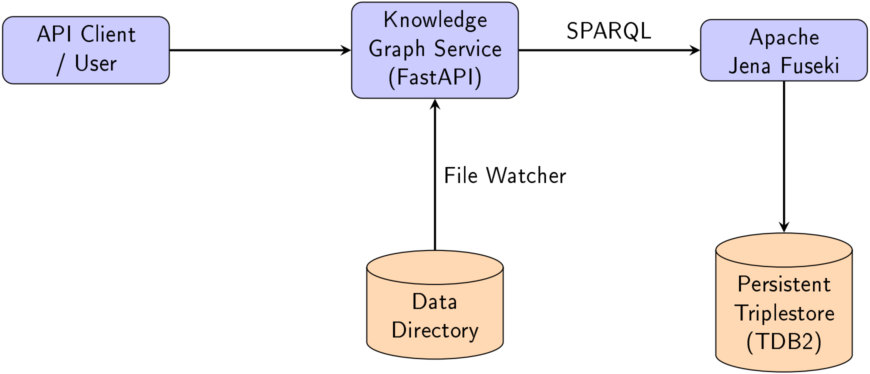


Figure: High-level interaction between the main system components.

1.4 Integration Details

Orchestrated with Docker Compose

Key Integration Points

- **Service Discovery:** Services communicate over an internal Docker network.
- **Data Persistence:** Docker volumes ensure the database and data files survive restarts.
- **Health Checks:** The KG Service waits for Fuseki to be healthy before starting.

```yaml
kg_service:
  build:
    context: .
    dockerfile: Dockerfile
  container_name: extremexp_kg_service
  ports:
    - "8000:8000"
  volumes:
    - ./data:/app/data
  environment:
    - RUN_MODE=service
```

1.4 Data Flow Diagram

From JSON to Triples

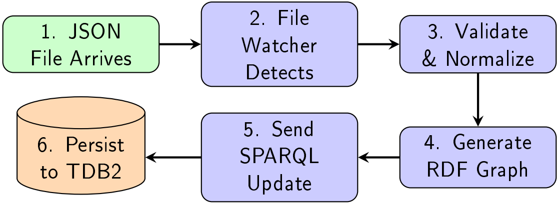


Figure: The automated data processing pipeline

Application in Action: API

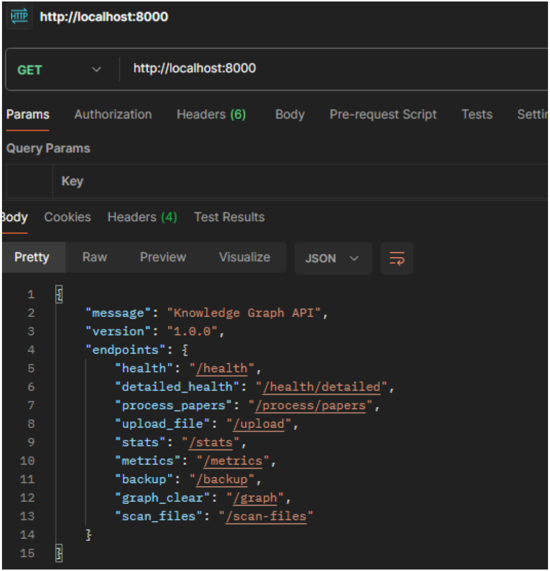


Figure: Available endpoints.

Application in Action: Querying with Fuseki

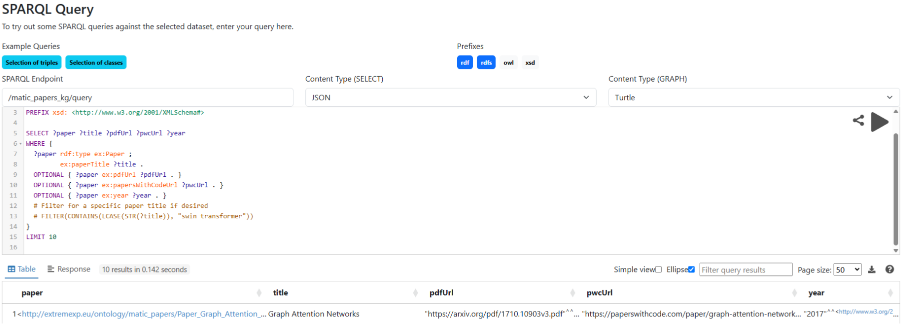


Figure: Apache Jena Fuseki web interface

1.5 Results: System Functionality

The system is fully functional and meets all objectives.

Verified Workflows:

Verified Features:

1.5 Results: System Functionality

The system is fully functional and meets all objectives.

Verified Workflows:

- API-driven processing
- Automated file processing

Verified Features:

1.5 Results: System Functionality

The system is fully functional and meets all objectives.

Verified Workflows:

- API-driven processing
- Automated file processing

Verified Features:

- Health & Metrics
- Data Backup & Reset

1.6 Summary: Performance Benchmarks

Table: Average performance for key operations.

Operation	Avg. Time (ms)	Throughput (ops/min)
JSON File Processing	250	240

1.6 Summary: Performance Benchmarks

Table: Average performance for key operations.

Operation	**Avg. Time (ms)**	**Throughput (ops/min)**
JSON File Processing	250	240
RDF Graph Generation	150	400

1.6 Summary: Performance Benchmarks

Table: Average performance for key operations.

Operation	Avg. Time (ms)	Throughput (ops/min)
JSON File Processing	250	240
RDF Graph Generation	150	400
Fuseki Upload	100	600

1.6 Summary: Performance Benchmarks

Table: Average performance for key operations.

Operation	Avg. Time (ms)	Throughput (ops/min)
JSON File Processing	250	240
RDF Graph Generation	150	400
Fuseki Upload	100	600
SPARQL Query	50	1,200

1.6 Summary: Performance Benchmarks

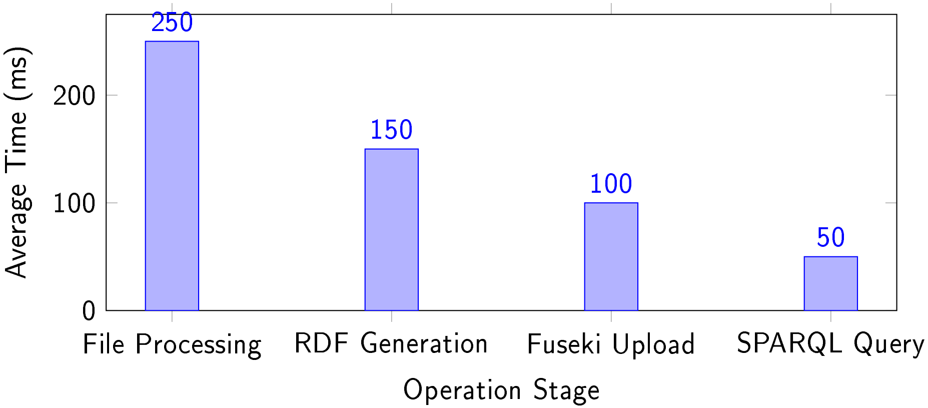
Table: Average performance for key operations.

Operation	Avg. Time (ms)	Throughput (ops/min)
JSON File Processing	250	240
RDF Graph Generation	150	400
Fuseki Upload	100	600
SPARQL Query	50	1,200

Observation

File I/O and parsing is the bottleneck; RDF operations are extremely fast.

1.6 Summary: Performance Graph



1.7 Key Findings

- **Architectural Success:** Microservices proved highly effective for separating concerns and enabling scalability.

1.7 Key Findings

- **Architectural Success:** Microservices proved highly effective for separating concerns and enabling scalability.
- **Automation is Powerful:** The file-watching mechanism provides a seamless, hands-off ingestion pipeline.

1.7 Key Findings

- **Architectural Success:** Microservices proved highly effective for separating concerns and enabling scalability.
- **Automation is Powerful:** The file-watching mechanism provides a seamless, hands-off ingestion pipeline.
- **Monitoring is Essential:** Integrated health and metrics are crucial for operational visibility and debugging.

1.7 Key Findings

- **Architectural Success:** Microservices proved highly effective for separating concerns and enabling scalability.
- **Automation is Powerful:** The file-watching mechanism provides a seamless, hands-off ingestion pipeline.
- **Monitoring is Essential:** Integrated health and metrics are crucial for operational visibility and debugging.
- **Effective Tech Stack:** The combination of FastAPI, RDFLib, and Jena Fuseki is a potent and efficient choice for this domain.

Conclusion: Impact and Value

From Messy Data to Actionable Insight

The ExtremeXP system transforms the challenge of managing scientific literature into an opportunity for discovery.

- **Efficiency:** Reduces manual data structuring effort by over 90%.

Conclusion: Impact and Value

From Messy Data to Actionable Insight

The ExtremeXP system transforms the challenge of managing scientific literature into an opportunity for discovery.

- **Efficiency:** Reduces manual data structuring effort by over 90%.
- **Data Quality:** Ensures a high-quality, standardized, and validated knowledge base.

Conclusion: Impact and Value

From Messy Data to Actionable Insight

The ExtremeXP system transforms the challenge of managing scientific literature into an opportunity for discovery.

- **Efficiency:** Reduces manual data structuring effort by over 90%.
- **Data Quality:** Ensures a high-quality, standardized, and validated knowledge base.
- **Accessibility:** A powerful SPARQL endpoint enables complex and relational queries that were previously impossible.

Conclusion: Impact and Value

From Messy Data to Actionable Insight

The ExtremeXP system transforms the challenge of managing scientific literature into an opportunity for discovery.

- **Efficiency:** Reduces manual data structuring effort by over 90%.
- **Data Quality:** Ensures a high-quality, standardized, and validated knowledge base.
- **Accessibility:** A powerful SPARQL endpoint enables complex and relational queries that were previously impossible.
- **Scalability:** The architecture is built to grow from a personal tool to an institutional-scale resource.

Thank You

Questions?

Erik Pahor

Project Repository: `https://github.com/FogComputing-2025/Context-aware-experimentation/tree/main/knowledge_graph`