Building Graph Search Components: From Query Patterns to LLM Integration

Abstract

This workshop explores building practical graph search components that make SPARQL accessible through intuitive interfaces. We'll dive into query pattern generation, front-end component design, and how to create embedding-ready outputs that work seamlessly with modern LLMs.

The Challenge

Graph databases are powerful, but SPARQL can be intimidating. Users want to search knowledge graphs like they search Google - with natural language and instant results. The challenge is bridging this gap while preserving the precision and expressiveness that makes SPARQL valuable.

Component Design

The Search Interface

Our component approach focuses on progressive query building. Users start with simple text input that gets transformed into structured queries behind the scenes. The interface provides:

- Auto-complete suggestions based on graph entities
- Visual query builders that generate SPARQL fragments
- Context-aware result previews
- One-click query refinement and expansion

Query Pattern Generation

The core innovation lies in smart pattern generation. Instead of forcing users to write SPARQL, we:

- Decompose complex queries into reusable fragments
- Generate template patterns that adapt to different graph structures
- Create embedding-compatible representations for each query pattern
- Enable natural language to SPARQL translation through LLM integration

LLM Integration Strategy

Modern graph search components need to speak both SPARQL and natural language. Our patterns are designed to be embedding-ready, allowing:

- Semantic similarity matching between user queries and graph patterns
- LLM-powered query explanation and suggestion
- Hybrid retrieval combining symbolic SPARQL execution with neural search
- Natural language result summarization

Real-World Implementation

Building this at enterprise scale taught us that users don't want to learn graph query languages - they want their questions answered. The component handles the complexity while presenting simple, familiar search interactions.

Key insights from production deployment:

- Users prefer guided search over free-form querying
- Visual feedback during query construction improves adoption
- Caching frequent patterns dramatically improves response times
- Integration with existing tools matters more than perfect graph coverage

Workshop Outcomes

Participants will reconstruct core components from scratch, learning how to:

- Build responsive search interfaces for graph data
- Generate reusable SPARQL query patterns
- Create embedding-read