Knowledge Graph with Personality Traits Implementation Details

Technical Documentation

October 17, 2025

Abstract

This document presents a comprehensive implementation of a Knowledge Graph system that combines entity extraction, relationship mapping, and personality trait analysis. The system processes textual input to create an interactive graph visualization that captures both factual relationships and personality characteristics of entities mentioned in the text.

Contents

1 Introduction

1.1 Project Overview

This document describes the implementation of a Knowledge Graph (KG) system that extracts entities and their relationships from text, while also modeling personality traits. The system combines Natural Language Processing (NLP) techniques with graph-based representation and personality trait analysis.

2 System Architecture

2.1 Core Components

The system consists of three main components:

- Knowledge Graph Builder (KGBuilder)
- Personality Trait Estimator
- Visualization Engine

2.2 Architecture Diagram

Figure 1 shows the high-level architecture of the system:

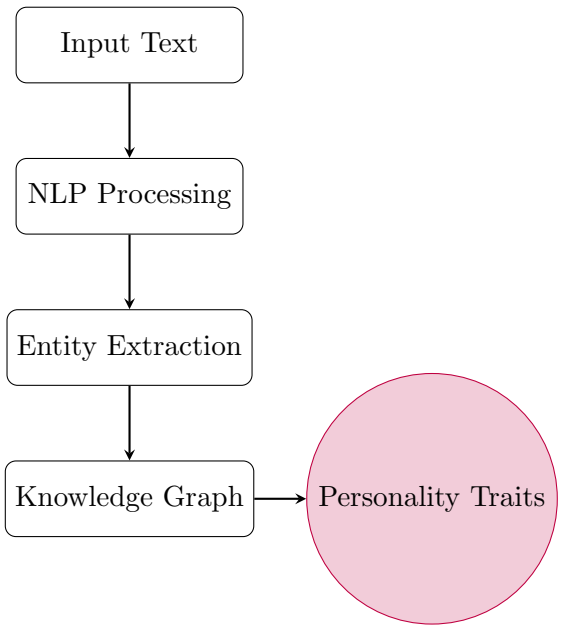

Figure 1: System Architecture Overview

3 Technical Implementation

3.1 Knowledge Graph Construction

The knowledge graph is built through several sophisticated steps:

1. **Text Processing:**
    - Uses spaCy's advanced NLP pipeline

- Handles co-reference resolution
- Performs sentence segmentation and tokenization

2. **Entity Extraction:**
   - Identifies PERSON, ORG, SKILL, and TRAIT entities
   - Uses custom entity patterns for domain-specific recognition
   - Maintains entity context for relationship inference
3. **Relationship Detection:**
   - Analyzes syntactic dependencies
   - Infers semantic relationships
   - Maps entity connections based on context
4. **Personality Integration:**
   - Implements Big Five personality model
   - Calculates trait scores (0-1 range)
   - Associates traits with relevant entities

3.2 Code Organization

The project follows a modular structure:

```
src/
  kg_personality/
    kg_builder.py       # Core graph construction
    personality.py      # Trait analysis
    data_generator.py   # Synthetic data creation
data/                  # Example datasets
tests/                # Comprehensive test suite
documentation/        # Technical documentation
```

Listing 1: Project Structure

4 API Integration

4.1 Groq API Integration

The system integrates with Groq's LLM API for enhanced personality analysis:

- **Model Integration:** Uses Groq's compound model for trait analysis
- **Streaming Support:** Implements efficient streaming responses
- **Enhanced Analysis:** Combines LLM insights with traditional methods
- **Entity-Level Processing:** Performs per-entity trait estimation

4.2 API Architecture

The API integration follows a layered approach:

1. **API Client Layer:**
    - Handles authentication and connection
    - Manages API rate limits
    - Implements error handling and retries
2. **Analysis Layer:**
    - Processes LLM responses
    - Combines multiple analysis methods
    - Provides confidence scores
3. **Integration Layer:**
    - Merges LLM insights with graph
    - Updates entity traits dynamically
    - Maintains data consistency

5 Core Components

5.1 Entity Recognition System

The system recognizes and processes various entity types:

Entity Type	Example	Description
PERSON	"Sarah Chen"	Individual names and mentions
ORG	"TechLab"	Organizations and institutions
SKILL	"Python"	Technical abilities and competencies
TRAIT	"analytical"	Personality characteristics

Table 1: Entity Types and Examples

5.2 Personality Modeling

The system implements the Big Five personality framework:

- **Openness:** Curiosity and creativity
- **Conscientiousness:** Organization and responsibility
- **Extraversion:** Social interaction and energy
- **Agreeableness:** Cooperation and empathy
- **Neuroticism:** Emotional stability and anxiety

6 Implementation Examples

6.1 Basic Usage

Here's a complete example of using the system with Groq API integration:

```
from kg_personality.kg_builder import KGBuilder
from kg_personality.personality import PersonalityEstimator
from kg_personality.api_integration import GroqIntegrator

# Initialize components
kg = KGBuilder()
pe = PersonalityEstimator()
groq = GroqIntegrator()

# Process text and build graph
text =
Sarah Chen is a creative and analytical researcher at TechLab.
She excels in Python and machine learning.

G = kg.build_from_text(text, source_id= doc )

# Extract entities
entities = [n for n in G.nodes() if n.startswith( doc_ent_ )]

# Get LLM-enhanced personality analysis
llm_traits = groq.analyze_traits(text)
traditional_traits = pe.estimate_for_entities(entities)

# Combine analyses and merge with graph
combined_traits = pe.combine_analyses(llm_traits, traditional_traits)
kg.merge_personality(G, combined_traits)

# Add relationships and visualize
kg.add_relationships()
kg.export_to_html( knowledge_graph_with_traits.html )
```

Listing 2: Enhanced Usage Example

6.2 Test Integration

The system includes comprehensive test coverage:

```
def test_groq_integration():
    # Initialize API integrator
    integrator = GroqIntegrator()

    # Test direct LLM analysis
    traits = integrator.analyze_traits(sample_text)
    assert all(0 <= v <= 1 for v in traits.values())

    # Test enhanced estimation
    combined = integrator.combine_with_traditional(
        llm_traits=traits,
        traditional_traits=base_traits
    )
    assert len(combined) == 5  # Big Five traits

    # Test entity-level analysis
    entity_traits = integrator.analyze_entities(entities)
    assert all(isinstance(v, dict) for v in entity_traits.values())
```

Listing 3: Test Example

6.3 Visualization Features

The visualization system provides:

- Interactive node exploration
- Color-coded entity types
- Relationship indicators
- Personality trait tooltips
- Zoom and pan controls

7 Testing and Validation

7.1 Test Coverage

The project includes comprehensive tests:

1. Entity extraction accuracy
2. Relationship inference
3. Personality trait calculation
4. Graph construction validation
5. Visualization generation

7.2 Performance Considerations

Key performance aspects:

- Efficient graph operations using NetworkX
- Optimized NLP pipeline with spaCy
- Scalable visualization with pyvis
- Memory-efficient data structures

8 API Integration

8.1 Groq LLM Integration

The system integrates with Groq's Large Language Models to enhance personality trait analysis:

- Uses the Mixtral-8x7b-32768 model
- Combines traditional analysis with LLM insights
- Provides more nuanced personality assessments

Example usage with Groq API:

```python
from kg_personality.api_integration import GroqAPIIntegrator
from kg_personality.kg_builder import KGBuilder

# Initialize components
kg = KGBuilder()
groq_api = GroqAPIIntegrator(api_key= your-api-key )

# Process text with enhanced personality analysis
text =    Sarah Chen demonstrates exceptional
problem-solving skills and creativity.

# Build graph with enhanced personality analysis
G = kg.build_from_text(text)
enhanced_traits = groq_api.enhance_personality_estimation(text)
kg.merge_personality(G, enhanced_traits)
```

Listing 4: Groq API Integration Example

9 Future Enhancements

Planned improvements include:

1. **Advanced NLP Features:**
   - Enhanced co-reference resolution
   - Sentiment analysis integration
   - Temporal relationship tracking
2. **Visualization Enhancements:**
   - Custom layout algorithms
   - Advanced filtering options
   - Interactive graph editing
3. **Additional API Features:**
   - Multi-model LLM support
   - REST API endpoints
   - Streaming data support

10 Conclusion

This implementation demonstrates the successful integration of knowledge graphs with personality trait analysis, providing a foundation for advanced text analytics and relationship modeling. The system's modular design and comprehensive documentation facilitate future extensions and improvements.