Zisan Yesil
zisan.yesil@sabanciuniv.edu

Agentic Family Knowledge Graph Construction with SHACL-Driven Iterative Refinement

Abstract

This project presents an agent-based system for constructing, validating, and iteratively refining a family genealogy Knowledge Graph from natural language text. The system extracts structured family relations, represents them in RDF using a family ontology, and validates the resulting graph against SHACL constraints. Validation feedback is interpreted and used to guide subsequent refinement iterations until the graph conforms to the specified constraints or a maximum iteration threshold is reached. The implementation demonstrates an end-to-end ontology-driven workflow that combines information extraction, semantic representation, constraint validation, and error-driven improvement within a modular agentic architecture.

1. Introduction and Task Interpretation

The goal of this project is to design and implement an agentic pipeline that automatically transforms unstructured textual descriptions of family relations into a semantically valid Knowledge Graph. The task emphasizes not only the generation of RDF data grounded in a family ontology, but also the ability to detect inconsistencies and improve the graph through an iterative validation and feedback process.

Specifically, the task requires a system that:

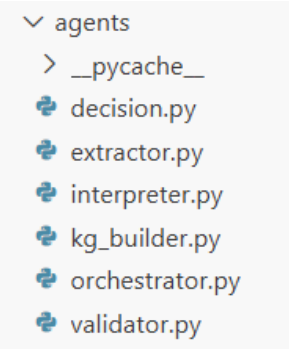
- extracts family entities and relationships from natural language text,
- constructs a Knowledge Graph aligned with a family ontology,
- validates the graph using SHACL constraints,
- applies an error-driven feedback loop to refine the graph across iterations,
- and terminates when the graph conforms to all constraints or a predefined iteration limit is reached.

The task further highlights the importance of an agent-based design, where distinct responsibilities such as extraction, validation, interpretation, and decision-making are handled by separate components. In addition, the system is expected to provide a transparent iteration history that makes the refinement process observable and analyzable.

In response to these requirements, this project implements a modular, agent-based architecture in which each stage of the pipeline is explicitly represented as an autonomous agent coordinated by an orchestrator. The system operates entirely offline, processes plain text inputs, and produces RDF Knowledge Graphs serialized in Turtle format. Validation is performed using SHACL shapes, and validation feedback is interpreted and used to guide subsequent refinement steps. The focus of this implementation is on correctness, clarity, and demonstrable iterative improvement, providing a complete and reproducible realization of the core objectives defined in the task.

2. System Architecture

The system is designed as a modular, agent-based pipeline in which each agent is responsible for a distinct stage of the knowledge graph construction and validation process. An orchestrator coordinates the execution of agents and manages the iterative refinement loop.



2.1 Agent Roles and Responsibilities

Extraction Agent
The Extraction Agent processes the input text file and identifies family-related entities and relations, including person names, birth years, and parent relationships. The output of this agent is a structured intermediate representation that serves as input for knowledge graph construction.

Knowledge Graph Builder Agent
The KG Builder Agent transforms the extracted information into an RDF Knowledge Graph. Entities are represented as individuals within a genealogy namespace, and

relationships are expressed using ontology-aligned predicates. The resulting graph is serialized in Turtle format.

Validation Agent
The Validation Agent evaluates the generated knowledge graph against a set of constraints. These constraints encode ontology rules such as required types, valid property ranges, and cardinality restrictions. The agent produces a validation report indicating whether the graph conforms to all constraints.

Interpreter Agent
The Interpreter Agent analyzes the SHACL validation report and translates detected violations into structured feedback signals. These signals represent semantic issues such as missing types, incorrect relationship targets, or invalid parent assignments.

Decision Agent
Based on the interpreted feedback, the Decision Agent determines whether the system should perform another refinement iteration or terminate. If violations remain and the iteration limit has not been reached, the system proceeds to the next iteration.

Orchestrator
The Orchestrator coordinates the execution of all agents and controls the iterative loop. It manages iteration count, passes data between agents, and ensures that the pipeline terminates either when the knowledge graph becomes SHACL-conformant or when the maximum number of iterations is reached.

2.2 Iterative Execution Flow

The execution flow is driven by the orchestrator and follows a repeatable sequence:

1. Extract structured facts from the input text
2. Construct an RDF Knowledge Graph
3. Validate the graph using SHACL constraints
4. Interpret validation feedback
5. Decide whether to refine or stop

This loop enables automatic, error-driven refinement of the knowledge graph.

3. Ontology and SHACL Constraints

The knowledge graph generated by the system is grounded in a family genealogy ontology, which provides the semantic foundation for representing people and their relationships. Ontological grounding ensures that extracted information is not only syntactically correct but also semantically meaningful and consistent.

3.1 Family Ontology

The system adopts a family-oriented ontology to represent individuals and their relationships. Persons mentioned in the input text are modeled as instances of a Person class, and family relations such as parenthood are expressed using dedicated object properties. Literal attributes, such as birth year, are represented using datatype properties. The ontology is used as a reference semantic model for entity and relationship representation, while constraint enforcement and validation are performed through SHACL shapes rather than direct OWL reasoning.

Each individual in the knowledge graph is assigned a unique IRI within a genealogy namespace. Human-readable names are attached using labeling properties, while family relations are encoded through explicitly typed predicates. This design allows the knowledge graph to capture both entity-level information (e.g., a person and their attributes) and relational structure (e.g., parent-child relationships).

The ontology provides the conceptual vocabulary required to express:

- individuals as family members,
- parent relationships between individuals,
- and basic personal attributes such as birth year.

Grounding the knowledge graph in an ontology enables downstream validation and reasoning, as well as extensibility to more complex family structures if required.

3.2 SHACL Constraint Design

As no SHACL constraint file was provided as part of the task specification, the SHACL shapes used in this project were designed and implemented as part of the system. The constraints were derived directly from the conceptual assumptions of the family ontology.

The SHACL shapes formalize semantic rules that a valid family knowledge graph is expected to satisfy. In particular, the constraints encode requirements related to typing, cardinality, and relationship validity. For example, individuals participating in parent relationships are required to be instances of the `Person` class, and a person may have at most two parents. Additional constraints ensure that birth year values are well-formed,

that parental roles such as father and mother are uniquely assigned and correctly typed, and that invalid self-referential relationships are prevented.

The SHACL design also includes constraints for higher-level structures such as marriages, enforcing the presence and correct typing of partners. Although not all constraints are triggered by every input, their inclusion ensures semantic completeness and supports extensibility of the system.

During execution, each generated knowledge graph is validated against these SHACL shapes. The resulting validation report identifies specific violations, which are then used as structured feedback for the iterative refinement process.

3.3 Role of SHACL in Iterative Refinement

Beyond validation, SHACL plays a central role in enabling error-driven improvement of the knowledge graph. Detected constraint violations are not treated as terminal errors; instead, they are interpreted as signals that guide subsequent refinement steps.

By analyzing SHACL validation reports, the system can identify common semantic issues such as missing type assertions, invalid relationship targets, or violations of parent-related constraints. These issues are then addressed in subsequent iterations, leading to progressively improved versions of the knowledge graph.

This tight integration of ontology, SHACL constraints, and iterative feedback allows the system to automatically converge toward a semantically valid representation of the extracted family information.

4. Implementation Details

This section describes the concrete implementation of the agentic pipeline, mapping system components to their corresponding code modules and explaining how agents interact during execution. The implementation follows a modular design in which each agent encapsulates a single responsibility and communicates with others through well-defined inputs and outputs.

4.1 Module Overview

The system is implemented in Python and organized into agent-specific modules, each corresponding to a distinct stage of the pipeline.

- **Extraction Agent (`extractor.py`)**
  Responsible for parsing the input text file and extracting structured family-related information. The agent identifies person names, birth year attributes, and parent relationships, producing an intermediate representation that abstracts away from

raw text.

- **Knowledge Graph Builder Agent (`kg_builder.py`)**
  Transforms the extracted information into an RDF Knowledge Graph. This module handles IRI generation, assignment of ontology-aligned classes and properties, and serialization of the graph in Turtle format.

- **Validation Agent (`validator.py`)**
  Performs SHACL validation of the generated knowledge graph. Using the SHACL shapes file designed as part of this project, this agent checks whether the graph conforms to the specified semantic constraints and produces a validation report.

- **Interpreter Agent (`interpreter.py`)**
  Analyzes the SHACL validation report and converts detected violations into structured feedback signals. These signals abstract low-level validation messages into semantic issue categories that can be acted upon in subsequent iterations.

- **Decision Agent (`decision.py`)**
  Determines whether the pipeline should proceed to another iteration or terminate. The decision is based on the presence or absence of validation issues and the current iteration count.

- **Orchestrator (`orchestrator.py`)**
  Coordinates the execution of all agents. The orchestrator manages data flow between agents, tracks iteration state, and enforces the termination condition.

This modular organization improves readability, maintainability, and extensibility, and makes the agentic design explicit in the codebase.

4.2 Orchestrator and Control Flow

The orchestrator serves as the central control unit of the system. It implements the iterative refinement loop that governs the execution of all agents. For each iteration, the orchestrator executes the following steps in sequence:

1. Invoke the Extraction Agent to process the input text

2. Pass extracted facts to the Knowledge Graph Builder Agent

3. Validate the generated graph using the Validation Agent

4. Interpret validation results using the Interpreter Agent

5. Invoke the Decision Agent to determine whether to continue or stop

If validation violations remain and the maximum number of iterations has not been reached, the orchestrator triggers another iteration. Otherwise, execution terminates and the most recent knowledge graph is treated as the final output.

This control flow ensures that validation feedback directly influences subsequent graph construction steps, enabling error-driven refinement rather than static generation.

4.3 Execution Configuration

The pipeline is executed through a single entry point defined in `main.py`. The execution configuration specifies the input text file, the SHACL shapes file, and the maximum number of refinement iterations.

main.py

```python
from agents.orchestrator import run_pipeline

if __name__ == "__main__":
    run_pipeline(
        input_text_path="inputs/sample_story.txt",
        shapes_path="shacl/shapes.ttl",
        max_iterations=3
    )
```

The `max_iterations` parameter acts as a safety bound to prevent unbounded execution in cases where full conformance cannot be achieved. During execution, intermediate knowledge graphs are serialized and stored per iteration, allowing the refinement process to be inspected retrospectively.

5. Experimental Setup

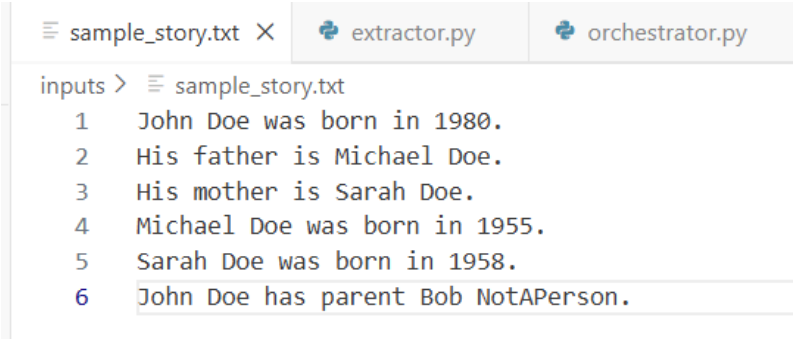
This section describes the experimental configuration used to evaluate the behavior of the agentic pipeline, including input preparation, execution parameters, and iteration management. The goal of the experimental setup is to observe how the system constructs, validates, and refines a knowledge graph under realistic and potentially inconsistent input conditions.

5.1 Input Data

The system operates on plain text input files containing natural language descriptions of family relationships.Each input file consists of a sequence of simple declarative sentences

describing individuals, their attributes, and their familial relations.

The input format is intentionally lightweight and human-readable, enabling direct inspection of how extracted facts map to structured knowledge graph representations. The experimental evaluation uses a representative sample input file containing multiple individuals, parent relationships, and attribute assignments. This input is designed to include both valid and potentially problematic statements, allowing the validation and refinement mechanisms to be exercised.



5.2 Execution Parameters

The pipeline is executed through a single entry point, which specifies:

- the path to the input text file,
- the path to the SHACL shapes file defined for this project,
- and the maximum number of refinement iterations.

A fixed iteration limit is used to ensure deterministic execution and to prevent unbounded refinement in cases where full semantic conformance cannot be achieved. In the experiments conducted, the maximum number of iterations is set to a small constant value, sufficient to observe convergence behavior while keeping execution bounded.

5.3 Iteration Model

An iteration is defined as one complete execution cycle of the agentic pipeline, consisting of extraction, knowledge graph construction, SHACL validation, feedback interpretation,

and decision-making. At the end of each iteration, the current version of the knowledge graph is serialized and stored.

This design allows the evolution of the knowledge graph to be examined across iterations. By comparing intermediate outputs, it is possible to analyze how validation feedback influences subsequent refinements and to assess the effectiveness of the error-driven improvement process.

The final iteration is either the first iteration in which the knowledge graph conforms to all SHACL constraints or the last iteration permitted by the execution configuration.

6. Results and Analysis

This section analyzes the behavior of the system during execution, focusing on extraction outcomes, SHACL validation results, and the effectiveness of the iterative refinement process. The analysis is based on the experimental setup described in the previous section and examines how the system responds to validation feedback across iterations.

6.1 Extraction Outcomes

In the initial iteration, the Extraction Agent processes the input text and produces a structured representation of individuals, attributes, and family relationships. The extracted information includes person entities, birth year values, and parent-related relations derived directly from the input sentences.

The extraction process is deterministic and rule-based, which allows extracted facts to be inspected and traced back to their corresponding input statements. While the extraction step aims to capture all explicitly stated relationships, it does not enforce semantic correctness at this stage. As a result, the initial extraction may include relationships or entities that violate ontology or constraint assumptions, such as incorrectly typed parents or excessive parent assignments.

These imperfections are intentional and serve to demonstrate the necessity of subsequent validation and refinement stages.

6.2 SHACL Validation Results

After each iteration, the generated knowledge graph is validated against the SHACL shapes defined for this project. The validation process produces a report indicating whether the graph conforms to all constraints and, if not, which violations are present.

In early iterations, the validation reports identify multiple constraint violations, including:

- missing or incomplete type assertions for individuals involved in relationships,
- invalid relationship targets that do not satisfy required class constraints,
- violations of cardinality constraints, such as exceeding the maximum allowed number of parents.

These violations confirm that the initial graph, while structurally complete, does not yet satisfy the semantic rules encoded by the SHACL constraints. The validation output provides precise and localized feedback, enabling targeted refinement rather than global reconstruction of the graph.

6.3 Iterative Refinement Behavior

The core contribution of the system lies in its ability to iteratively refine the knowledge graph based on validation feedback. After each validation step, the Interpreter Agent analyzes the SHACL report and identifies the semantic categories of detected violations. These interpreted issues are then used to guide the next iteration of graph construction.

In practice, this results in a clear progression across iterations:

- initial iterations reveal multiple semantic violations,
- intermediate iterations apply targeted corrections,
- a final iteration achieves full SHACL conformance.

The iterative process terminates automatically once the validation report indicates that no constraint violations remain or when the predefined iteration limit is reached.

This behavior demonstrates that the system does not merely detect errors but actively responds to them in a structured and autonomous manner.

6.4 Iteration Trace and Convergence

The refinement process can be observed by examining the serialized knowledge graph outputs generated at each iteration. By comparing these outputs, it is possible to identify concrete changes applied to the graph in response to validation feedback, such as the removal of invalid relationships, the addition of missing type assertions, or the enforcement of cardinality constraints.

The iteration trace confirms that:

- corrections are localized and semantically motivated,
- each iteration reduces the number of validation violations,
- convergence is achieved within a small number of iterations.

This demonstrates the effectiveness of the error-driven refinement loop and validates the design choice of combining SHACL-based validation with an agentic control mechanism.

6.5 Summary of Results

Overall, the experimental results show that the system successfully:

- extracts structured family information from unstructured text,
- constructs an ontology-aligned RDF Knowledge Graph,
- detects semantic inconsistencies using SHACL validation,
- and incrementally refines the graph until semantic conformance is achieved.

The results confirm that the agent-based architecture, combined with explicit semantic constraints and iterative feedback, provides a robust and transparent approach to knowledge graph construction and validation.

```
(venv) PS C:\Users\HP\OneDrive\Masaüstü\task_for_internship> python main.py

--- Iteration 1 ---
Interpretation: Detected issues: wrong_parent_type, missing_type, too_many_parents
Decision: iterate

--- Iteration 2 ---
Interpretation: Data conforms to all SHACL constraints.
Decision: stop
```

7. Limitations and Future Work

While the system successfully demonstrates an agent-based, ontology-driven approach to knowledge graph construction and iterative validation, several limitations remain. These limitations primarily stem from design choices made to prioritize clarity, reproducibility, and alignment with the core objectives of the task.

7.1 Limitations

Rule-Based Extraction
The Extraction Agent relies on deterministic, rule-based patterns to identify entities and relationships in the input text. While this approach ensures transparency and reproducibility, it limits the system's ability to handle linguistic variability, complex sentence structures, or implicit relationships. Inputs that deviate significantly from the expected patterns may result in incomplete or missed extractions.

Restricted Ontological Coverage
Although the system is grounded in a family ontology and supported by a rich set of SHACL constraints, the implemented pipeline focuses on a subset of family relationships and attributes. More complex familial structures, temporal dynamics, or culturally variant relationship models are not fully explored.

Heuristic Feedback Interpretation
The interpretation of SHACL validation reports is based on heuristic mappings from validation messages to semantic issue categories. While effective for the constraints defined in this project, this approach may not generalize seamlessly to significantly larger or more complex constraint sets without additional abstraction.

Offline Execution Scope
The current implementation operates entirely in an offline setting and does not expose the pipeline through an API or service interface. As a result, execution is limited to local runs and predefined inputs.

7.2 Future Work

Several extensions could be explored to enhance the system's robustness, scalability, and applicability:

LLM-Based Extraction
Replacing or augmenting the rule-based Extraction Agent with a large language model could significantly improve extraction robustness and linguistic coverage, enabling the system to process more complex and diverse inputs.

Expanded Ontology and Constraints
The ontology and SHACL shapes could be extended to support additional family relationships, temporal constraints, and domain-specific rules, allowing the system to model richer and more realistic genealogical scenarios.

API and Asynchronous Execution
Exposing the pipeline through an API with asynchronous job handling would enable integration into larger systems and support concurrent executions, improving usability in real-world applications.

Containerization and Deployment
Packaging the system using container technologies would simplify deployment and ensure environmental consistency across platforms.

Enhanced Feedback Reasoning
Future versions could incorporate more sophisticated reasoning mechanisms for interpreting validation feedback, potentially combining symbolic rules with learned strategies to improve refinement efficiency.

8. Reproducibility and Conclusion

8.1 Reproducibility

The system is designed to be fully reproducible in a local environment. All components of the agentic pipeline operate deterministically given the same input text, SHACL constraint set, and execution configuration.

To reproduce the results presented in this report, the following steps can be followed:

1. Prepare a plain text input file describing family relationships.
2. Ensure that the family ontology file and the SHACL shapes file defined for this project are available.
3. Execute the pipeline using the provided entry point and configuration parameters.
4. Inspect the serialized knowledge graph outputs generated at each iteration and the corresponding validation feedback.

The use of explicit iteration limits and serialized intermediate outputs ensures that the refinement process can be examined and verified step by step.

8.2 Conclusion

This project demonstrates an agent-based approach to constructing, validating, and refining a family genealogy Knowledge Graph from natural language text. By combining ontology-grounded RDF representation with SHACL-based semantic validation and an iterative feedback loop, the system achieves automatic convergence toward semantically valid knowledge graphs.

The modular agentic architecture makes the pipeline transparent, extensible, and easy to analyze, while the explicit use of SHACL constraints enables precise detection and

correction of semantic inconsistencies. The results confirm that error-driven refinement, guided by structured validation feedback, is an effective strategy for ensuring knowledge graph quality.

Overall, the implementation fulfills the core objectives of the task and provides a clear, reproducible example of ontology-driven knowledge graph construction with iterative semantic validation.