UNIVERSITÉ CÔTE D'AZUR | ÉCOLE UNIVERSITAIRE DE RECHERCHE SYSTÈMES NUMÉRIQUES POUR L'HUMAIN

Semantic Similarity Search

A report of current work on ECP sheets generation

Master 1 Computer Science

Academic Year 2025–2026

Student: Belhout Oussama

Supervisor: Pr.Nadia Abchiche-Mimouni

Contents

Abstract

This project implements a semantic search engine designed for the "Education for a Culture of Peace" (ECP). Unlike traditional keyword search engines that rely on simple text matching, the domain of Peace Pedagogy holds rich semantic attributes—inherent to its philosophical nature—which require advanced **Knowledge Representation (KR)** technologies to be fully captured. The system models the pedagogical "Anatomy" of a lesson—its tools, virtues, and peace axes—using a formal **Ontology**, and calculates relevance using a multi-dimensional **Semantic Similarity** algorithm. This report details the theoretical foundations of the system and its technical architecture.

The primary resource for the theoretical framework of this work was the lecture material from the "Knowledge Representation and Reasoning" course by Professor Khellaf (USTHB University), available in the project documentation.

1 Part I: Theoretical Foundations

Before analyzing the implementation, it is essential to understand the two pillars of Knowledge Representation used in this project: **Semantic Networks** and **Ontologies (Description Logic)**.

1.1 Semantic Networks:

In computer science, a **Semantic Network** is a graphical representation of knowledge, comparable to a mind map.

- **The Structure:**
  - **Nodes:** Represent concepts (e.g., "Meditation", "Science", "Child").
  - **Edges (Arcs):** Represent relationships between nodes (e.g., *is_a*, *uses*, *part_of*).
- **The Logic of Similarity:** In a semantic network, meaning is defined by connectivity. Two concepts are considered similar if they share the same "neighbors" in the graph.
  Example: If *Lesson A* is connected to "Creativity" and "Group Work", and *Lesson B* is also connected to these nodes, the network considers them semantically close, even if their titles differ.
- **Relevance to Project:** This project uses this graph-based intuition to calculate similarity. It treats every lesson as a node and measures how many neighbors (Tools, Virtues) it shares with other lessons.

1.2 Ontologies and Description Logic:

While Semantic Networks are intuitive, they lack rigor. **Ontologies** provide the necessary structure based on **Description Logic (DL)**, a decidable fragment of First-Order Logic.

- **The TBox (Terminological Box):** This is the "schema" or blueprint that defines the vocabulary and rules.
  - *Classes (Sets):* e.g., "Every `Lesson` is an `EducationalComponent`."
  - *Properties (Roles):* e.g., "A `Lesson` must have a relationship `hasAxis` pointing to a `PeaceAxis`."
- **The ABox (Assertional Box):** This contains the data itself—the specific instances that populate the TBox.
  Instance Example: "`Mandala_Activity` is an instance of the class `Lesson`."
- **Relevance to Project:** i use the **Web Ontology Language (OWL)** to enforce strict typing.

1.3 The Hybrid Approach

This project tries to merges these two paradigms:

1. **Storage:** Uses **Ontology (DL)** to ensure data integrity and formal classification.
2. **Calculation:** Uses **Semantic Network** principles (Set Theory/Graph traversal) to compute similarity scores efficiently.

2 Part II: Technical Solution and Architecture

The solution is composed of four distinct modules: Knowledge Modeling, Data Ingestion, Population, and the Similarity Engine.

2.1 Knowledge Modeling (The Ontology)

File Reference: `src/ontology_builder.py`

To this date, this are the deduced foundations of the system knowledge, it is ontology defined using the `owlready2` library. This corresponds to the **TBox**.

- **Class Hierarchy:** The code defines a clear taxonomy where `EducationalComponent` and `PeaceComponent` act as root classes. Specific subclasses formally differentiate between `Tool` (e.g., Meditation), `Strategy` (e.g., Cooperative Learning), `Domain` (e.g., Sciences), and `Virtue` (e.g., Empathy).
- **Relational Properties:** Semantic links are defined via `ObjectProperties`:
  - `hasAxis`: Links a Lesson to the fundamental axes (Self, Others, Environment).
  - `developsVirtue` and `usesTool`: Map the pedagogical intent of a lesson.

2.2 Data Ingestion (The Parser)

File Reference: `src/pdf_parser.py`

To populate the ontology, the system extracts structured data from unstructured sources (PDF pedagogical sheets).

- **Metadata Extraction:** The parser extracts explicit data like `target_age_min` and `domain` from filenames and document headers.
- **Semantic Extraction:** It uses keyword detection dictionaries to infer semantic tags. For instance, detecting "mindfulness" in the text automatically tags the lesson with the tool `meditation`, converting raw text into ontology-compliant concepts.

2.3 Population (The Data Loader)

File Reference: `src/data_loader.py`

This module bridges the parser and the ontology, converting JSON data into the **ABox** (Ontology Instances).

- **Normalization:** Strings are cleaned to create valid ontology identifiers (URIs).
- **Instantiation:** The system creates individual `Lesson` instances and dynamically links them to existing TBox classes (`PeaceAxis`, `Tool`) using a `search_one` method to prevent duplication.

2.4 The Intelligence (Similarity Engine)

File Reference: `src/similarity_engine.py`

This is the core algorithmic component. Rather than relying on a slow logical reasoner, it implements a custom weighted set-similarity algorithm (Semantic Network approach).

2.4.1 Multi-Dimensional Similarity

The system aggregates scores from 7 different dimensions, weighted by pedagogical importance:

- **Peace Axes (25%):** The most critical pedagogical component.
- **Tools (20%) & Virtues (20%):** The "how" and "why" of the lesson.
- **Secondary Factors:** Strategies, Age, Duration, and Domain.

2.4.2 The Mathematical Model (Jaccard Index)

For categorical data (Axes, Tools, Virtues), the system uses the **Jaccard Similarity** coefficient:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{1}$$

This measures the intersection over the union. If *Lesson A* uses {Art, Meditation} and *Lesson B* uses {Art, Sport}:

- Intersection ($A \cap B$): {Art} (Size: 1)
- Union ($A \cup B$): {Art, Meditation, Sport} (Size: 3)
- Similarity: $1/3 \approx 0.33$

For continuous data (Age, Duration), the system implements compatibility functions (e.g., ratio of overlaps) rather than strict equality.

2.5 The User Interface (Query Engine)

File Reference: `src/query_engine.py`

This module abstracts the complexity of OWL for the end-user.

- **Temporary Instantiation:** When a user queries (e.g., "Find lessons about Nature"), a `temp_query_lesson` is created inside the ontology.
- **Comparison & Explainability:** This temporary instance is compared against the database. Crucially, the system returns a **breakdown** explaining *why* a lesson is similar (e.g., "Shared Virtues: Gratitude, Empathy"), providing transparency to the AI's decision-making.

3 Conclusion

This project demonstrates a sophisticated application of Knowledge Engineering. By formalizing **Peace Pedagogy** into an **Ontology**, the system transcends simple text search. It understands that a lesson using "Mindfulness" is structurally similar to one using "Meditation" because they share the same ontological class. The robust architecture—strictly defined by Description Logic in storage but flexible in retrieval using Semantic Network principles—ensures both data integrity and high-quality search results.