Lab Assignment
Discovering Horn Rules

Nicoleta Preda
nicoleta@preda.fr

January 24, 2025

1 How to Evaluate Rules Using an SQL Interface

The goal of this section is to understand how to translate the computation of the *support* and *confidence* metrics of a rule using an SQL interface. We consider a generic rule R of the form:

$$R: \quad B(\alpha, \beta) \Rightarrow r(\alpha, \beta)$$

where:

- $B(\alpha, \beta)$ represents the body of the rule, which can include one or more conditions.
- $r(\alpha, \beta)$ corresponds to the head relation of the rule.

1.1 Metric Formulas

The two main metrics used to evaluate a rule are defined as follows:

Support: The support of a rule R measures the number of facts in the knowledge base $\mathcal{K}$ that validate the rule R. Mathematically, it is expressed by the following formula:

$$\mathit{support}(R) = |\{p : (\mathcal{K} \wedge R \models p) \wedge p \in \mathcal{K}\}|$$

Confidence: The confidence of a rule R represents the proportion of correct predictions among all predictions made by the rule. It is given by:

$$\mathit{confidence}(R) = \frac{\mathit{support}(R)}{\mathit{support}(R) + |\mathit{cex}(R)|}$$

where:

- $\mathit{support}(R)$ is the number of correct predictions (*true positives*).
- $\mathit{cex}(R)$ represents the counterexamples, i.e., predictions made by R that are not valid in the knowledge base $\mathcal{K}$.

Note: Confidence metrics vary depending on the adopted hypothesis. Therefore, we define:

- *owa-conf*: Confidence based on the open-world assumption.
- *cwa-conf*: Confidence based on the closed-world assumption.
- *pca-conf*: Confidence based on the partial completeness assumption.

1.2 Exercise: Evaluating a Rule on Wikidata

Objective: Apply the concepts of *support* and *confidence* to evaluate a rule based on data from Wikidata. Consider the following rule:

$$R: \quad \texttt{positionHeld}(x, \texttt{UKPrimeMinister}) \Rightarrow \texttt{memberOf}(x, \texttt{BullingdonClub})$$

Listing 1: UK Prime Ministers Who Were Not Members of the Bullingdon Club

```

SELECT DISTINCT ?primeMinister ?primeMinisterName
WHERE {
  ?primeMinister p:P39 ?statement.
  ?statement ps:P39 wd:Q14211.         # Position: UK Prime Minister
  FILTER NOT EXISTS {
    ?primeMinister wdt:P463 wd:Q469039. # Not a member of Bullingdon Club
  }
  OPTIONAL {
    ?primeMinister rdfs:label ?primeMinisterName.
    FILTER(LANG(?primeMinisterName) = "en"). # Retrieve labels in English
  }
}
```

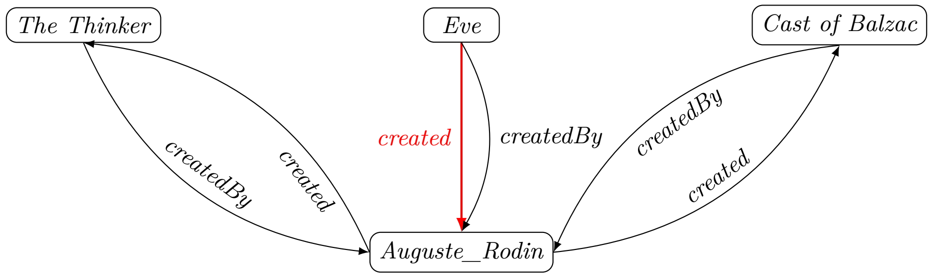
Testing Platforms The **Wikidata** and **YAGO** platforms provide online SPARQL endpoints:

- Wikidata SPARQL Endpoint
- YAGO SPARQL Endpoint

To explore the predicates available on Wikidata, consult the complete list of properties: Wikidata Property List.

2 Computing Metrics on a Given Graph

Consider the relations *created* and *createdBy*, along with the following knowledge graph:



The thick red line represents an erroneous fact, extracted incorrectly instead of its inverse. The domains of the two relations should normally be disjoint, as these relations are inverses of each other and are not symmetric. However, due to this error, the domains of *createdBy* and *created* share the element *Eve* in common.

1. Compute:

$$\textit{pca-conf}(\textit{createdBy}(x, y) \Rightarrow \textit{created}(x, y)) = ?$$

$$\textit{pca-conf}(\textit{created}(x, y) \Rightarrow \textit{createdBy}(x, y)) = ?$$

2. What do you observe?

3 Detection of Subproperty Relationships

In the context of description logic, the concept of a subproperty is essential for modeling hierarchical inclusion between properties. If a property r_1 is a subproperty of r_2 (denoted $r_1 \sqsubseteq r_2$), it means that all instances of r_1 are also instances of r_2. In other words, the subproperty relationship expresses that r_1 is a specialization of r_2, thereby formalizing an inheritance relationship between the two properties.

The subproperty relationship, denoted $r_1 \sqsubseteq r_2$, can be expressed as a Horn clause as follows:

$$r_1(x, y) \Rightarrow r_2(x, y)$$

This rule states that if a pair (x, y) satisfies the relation $r_1(x, y)$, then it must also satisfy the relation $r_2(x, y)$. In other words, every instance of the property r_1 implies a corresponding instance of the property r_2, thus modeling the hierarchical inclusion between the two properties.

Consider the following property:

Property 3.1 *Let r_1 and r_2 be two functional relations with $r_1 \sqsubseteq r_2$. Then,*

$$\textit{pca-conf}(r_1 \sqsubseteq r_2) = 1 \quad \textit{and} \quad \textit{pca-conf}(r_2 \sqsubseteq r_1) = 1.$$

4 Query Expressiveness

4.1 Objective: Comparing SPARQL and SQL

The goal of this section is to analyze the SQL equivalents of the OPTIONAL clause, FILTER NOT EXIST clause, and path patterns found in SPARQL queries. We assume that the knowledge base is stored in a single table, `KB`, with three attributes: `S` (subject), `P` (predicate), and `O` (object). This schema, which may seem inefficient, has proven to be highly performant. In particular, it is the schema adopted by the RDF3X system, which has demonstrated superiority compared to a partitioned schema where each predicate is stored in a separate table `P(S, O)`—`P` for predicate, `S` for subject, and `O` for object.

4.2 Exercises

Translate the following SPARQL queries into SQL equivalents for the OPTIONAL clause and path queries.

Listing 2: SPARQL Query 2

```
SELECT DISTINCT ?scientist
WHERE {
  ?scientist rdf:type yago:Scientist.
  FILTER NOT EXISTS {
    ?scientist schema:award yago:NobelPrize.
  }
}
```

Listing 3: SPARQL Query 2

```
SELECT ?pName ?date
WHERE {
  ?p schema:award yago:NobelPrize.
  ?p rdfs:label ?pName.
  OPTIONAL { ?p schema:birthDate ?date. }
}
```

Listing 4: SPARQL Query 3

```
SELECT ?pName
WHERE {
  ?p rdfs:label ?pName.
  ?p rdf:type yago:Scientist.
}
```

Listing 5: SPARQL Query 4

```
SELECT ?pName
WHERE {
  ?p rdfs:label ?pName.
  ?p rdf:type/rdfs:subClassOf* yago:Scientist.
}
```

4.3 Testing the Solutions

We suggest using the online SQL database tool: https://sqliteonline.com/.

Creating and Dropping the Table Before inserting data, we need to create the table. To avoid duplication errors, drop the table if it already exists using:

Listing 6: Dropping the Table

```
DROP TABLE IF EXISTS KB;
```

Listing 7: Creating the KB Table

```
CREATE TABLE KB (
  S TEXT,
  P TEXT,
  O TEXT
);
```

Inserting Data Insert the following data into the table:

Listing 8: Inserting Data into KB

```
INSERT INTO KB (S, P, O) VALUES
('yago:Marie_Curie', 'rdfs:label', '"Marie Curie"@en'),
('yago:Max_Planck', 'rdfs:label', '"Max Planck"@en'),
('yago:Marie_Curie', 'rdf:type', 'yago:Scientist'),
('yago:Dmitri_Mendeleev', 'rdf:type', 'yago:Scientist'),
('yago:Max_Planck', 'rdf:type', 'yago:TheoreticalPhysicist'),
('yago:TheoreticalPhysicist', 'rdfs:subClassOf', 'yago:Physicist'),
('yago:Physicist', 'rdfs:subClassOf', 'yago:Scientist'),
('yago:Marie_Curie', 'schema:birthDate', '"1867-11-07"^^xsd:date'),
('yago:Marie_Curie', 'rdfs:label', '"Marie Curie"@fr'),
('yago:Max_Planck', 'rdfs:label', '"Max Planck"@de'),
('yago:Marie_Curie', 'schema:award', 'yago:NobelPrize'),
('yago:Marie_Curie', 'schema:award', 'yago:AlbertMedal'),
('yago:Max_Planck', 'schema:award', 'yago:NobelPrize');
```

These rows represent RDF facts expressed as triples `(subject, predicate, object)`.

Verifying the Data After insertion, you can verify the table's content using:

Listing 9: Displaying Data from KB

```
SELECT * FROM KB;
```